An Area Efficient FPGA: Design using Non-Volatile RAM

by

Raffed Belal

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER ENGINEERING

May, 1993

Order Number 1354057

An area efficient FPGA: Design using non volatile RAM

Belal, Raffed, M.S.

King Fahd University of Petroleum and Minerals (Saudi Arabia), 1993

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

An Area Efficient FPGA: Design Using Non Volatile RAM

BY

RAFFED BELAL

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
In
COMPUTER ENGINEERING

May, 1993

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
DHAHRAN 31261, SAUDI ARABIA

COLLEGE OF GRADUATE STUDIES

This thesis is written by **RAFFED BELAL** under the direction of his Thesis Advisor and approved by Thesis Committee, has been presented to and accepted by the Dean of the College of Graduate Studies, in partial fulfillment of the requirements for the degree of MASTER of SCIENCE.

Thesis Committee

June 21, 1993

Dr. Alaaeldin A. M. Amin
Thesis Advisor

21/6 1993

Dr. Habib Youssef
Member

Dr. Muhammad S. T. Benten
Member

Dr. Rafik Braham
Member

21.6.93

Department Chairman
Dr. Samir H. Abdul-Jauwad

Dean, College of Graduate Studies
Dr. Ala Alrabeh

Date: 21-6-93

In Memory of my Father,
To my Mother, and
To my Fiancee

ACKNOWLEDGMENT

Acknowledgment is due to King Fahad University of Petroleum and Minerals for support .

I would like to express my deep appreciation to Professor Alaaeldin A. M. Amin and Professor Habib Youssef who served as my major advisor and my co-advisor respectively. Their careful guidance and supervision have made the completion of this work possible. Special thanks are due to Professors Muhammed S. T Benten and Rafik Braham for their helpful comments and suggestions. Finally, I wish to express my deep gratitude for Professor Samir H. Abdul-Jauwad, the department chairman for his encouragement and support.

TABLE OF CONTENTS

CHAPTER 4:THE INTERCONNECTION STRUCTURE

CHAPTER 5: THE CLOCK AND THE POWER DISTRIBUTION SYSTEMS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

NAME OF STUDENT: RAFFED BELAL

TITLE OF STUDY: AN AREA EFFICIENT FPGA: DESIGN USING NON VOLATILE RAM

MAJOR FIELD: COMPUTER ENGINEERING

DATE OF DEGREE: May, 1993

The use of Field Programmable Gate Arrays (FPGAs) for the design of Application Specific Integrated Circuits (ASICs) is becoming increasingly popular. This is due to their fast turn around time, flexible architecture and their well developed design automation tools. Compared to Mask Programmable Gate Arrays (MPGAs), FPGAs have the advantage of field programmability requiring only few minutes to be customized. FPGAs, however, are not as dense as MPGAs due to their programming area overhead. The architectural flexibility and gate density of FPGAs far surpasses those of the well known Electrically Programmable Logic devices (EPLDs).

Several technologies have been used for FPGAs customization. These include, SRAM, anti-fuse, and floating gate technologies. In contrast to other technologies, the SRAM-based FPGAs are volatile and hence the configuration data has to be loaded in the device every time the system is powered-up. Alternatively, anti-fuses are one time programmable lacking the necessary flexibility for design changes. A new floating gate technology combining the SRAM and the antifuse advantages was developed. The new technology uses a new nonvolatile SRAM cell. Compared to the standard SRAM-based FPGAs, the new technology speed increase by using a charge pump which causes a reduction of the interconnect switch resistance. Compared to antifuse-based FPGAs, the new technology is reprogrammable and fully testable. This results in high performance FPGA-implemented circuits.. This work reports the initial development of an architecture based on the non volatile cell. The basic logic module was designed, laid-out and verified through simulations. The interconnect architectural parameters were studied. A model was developed to estimate such parameters.

MASTER OF SCIENCE DEGREE

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
DHAHRAN, SAUDI ARABIA

May, 1993

خلاصة الرسالة

إسم الطالب : رافد بلال

عنوان الدراسة :تصميم مصفوفة بوابية قابلة للبرمجة باستخدام ذاكرة قابلة مستقرة

التخصص : هندسة الحاسب الآلي

تاريخ الشهادة :مايو ١٩٩٣

يزداد إستعمال المصفوفات البوابية القابلة للبرمجة في تصميم الدوائر التكاملة المقصورة على تطبيقات محمودة وذلك نظرا لما تتمتع به من سرعة في الدورة الزمنيةللتصميم ومرونة في تشكيل البنية الداخلية إضافة إلى تقدم وسائل التصميم الآلية .

و تنفرد المصفوفات البوابية القابلة للبرمجة بخاصية البرمجة الميدانية مما يسهل عملية تجهيزها وفق الطلب، بيدأن كثافتها أقل من مثلاتها في المصفوفات القابلة للبرمجة باستخدام اقنعة خاصة بسبب المساحة الاضافية المطلوبة لبرمجتها .و من جهة أخرى فإن المصفوفات البوابية القابلة للبرمجة تتفوق عن النبائط المنطقية المبرمجة كهربائيا في مرونة البنية الداخلية والكثافة البوابية، ولقد إستعملت تقنيات متعددة في تهيئة المصفوفات البوابية القابلة للبرمجة من بينها تقنية الذاكرة غير المستقرة ذات النفذ العشوائي وتقنية نظير المصهر وتقنية البوابات الطافية. خلافا للتقنيات الأخرى، فإنه ينبغي إعادة تحميل الذاكرة غير المستقرة ذات النفذ العشوائي بمعطيات التهيئة عند بدء التشغيل، وفي المقابل فان نظائر المصهر تفتقد الى المرونة المطلوبة عند تعديل التصاميم .

وإستنادا لما سبق فقد تم تطوير تقنية بوابة عائمة جديدة تجمع بين مزايا تقنية الذاكرة غير المستقرة ذات النفذ العشوائي وتقنية نظير المصهر وتعتمد على خلايا الذاكرة المستقرة ذات النفذ العشوائي. ولقد إستعملت في التقنية الجديدة مضخات شحن تعمل على خفض مقاومة مفاتيح الربط مما أدى إلى زيادة سرعة المصفوفات مقارنة بالمصفوفات المبنية على الذاكرة غير المستقرة ذات النفذ العشوائي، إضافة الى كونها مستقرة وتوفر أمنية المعلومات، كما تتمتع التقنية الجديدة بخاصية إعادة البرمجة والإختبار الكلي مقارنة بالمصفوفات ذات نظائر المصهر.

يقدم هذا البحث نتائج التطوير الاولي لبنية معتمدة على خلية مستقرة لقد تم تصميم وتخطيط الوحدة المنطقية الأساسية ، كما تم إختبارها باستعمال العاكمات اضافة الى دراسة التحولات البنيوية للربط ، وبناءا عليه تمت صياغة نموذج لتقدير هذه التحولات.

درجة الماجستير في العلوم

جامعة الملك فهد للبترول والمعادن

الظهران . المملكة العربية السعودية

التاريخ: مايو 1993

CHAPTER 1

INTRODUCTION

A brief description of the Application Specific Integrated Circuits (ASICS) will be given in the following sections. They are compared with the design methodologies based on SSI/MSI and LSI/VLSI. Field Programmable Gate Arrays (FPGAs) will be introduced and their basic components defined. Finally, the work done on a new non volatile FPGA is summarised. This chapter is concluded by the description of the organisation of this report.

1.1 ASICs Designs

There is no universally accepted method for classifying logic ICs, however, one widely accepted classification [Fey87] is shown in Figure 1.1. Logic functions can be implemented using either a combination of SSI and MSI parts or specially designed Application Specific Integrated Circuits (ASICs). In general, choosing one of these two approaches will depend on the gate density requirement and the production volumes needed. A determinant factor of the IC-related cost of a digital design is the number of

gates per pin. It has been found that designs using ASICs cost less than designs containing a combination of SSI/MSI and LSI/VLSI devices, provided that the number of gates per pin is 2-3 times that of the product containing standard devices [Fey87]. Since the number of gates per pin for ASICs is available in a wide range (typically 40-200 gates per pin), a proper ASIC chip can generally be chosen which results in a lower design cost. Figure 1.2 shows the cost-effective range of various ASIC design methodologies related to their gate densities and production volumes. For a successful ASIC design both a suitable technology and design methodology should be adopted taking into account the performance requirements as well as the anticipated volume and cost constraints.

FPGAs design methodology constitutes a revolutionary idea in semiconductor integrated circuits. This methodology reduces the turn around time (TAT) from 6-8 weeks required by mask programmable gate arrays to only few minutes. In addition, the design prototyping cost is cut from thousands of dollars to less than a hundred. FPGAs combine the architectural flexibility and efficiency of the Mask Programmable Gate Arrays (MPGAs) and the user friendliness of the Programmable Logic Devices (PLDs). FPGAs are particularly effective in rapid system prototyping and in-field testing. This allows early product introduction and a good market share. They have been successfully used in scientific, military, communications and in several other special digital systems [Fur90]. The whole glue logic of a micro computer can be replaced by an FPGA. Hence only the microprocessor and memory in addition to an FPGA are needed to build the kernel of a very compact micro-computer board. Finally, FPGAs have been used to build reconfigurable computer peripherals such as disk and

printer controllers. FPGAs are expected to be the third paradigm shift in electronics industry after semiconductor memories and microprocessors [East91].

1.2 Architecture of Field Programmable Gate Arrays

The architecture of an FPGA refers to its basic building blocks, the position of these blocks relative to each other and their interconnect strategy. In addition, it also takes into account the used programming technology.

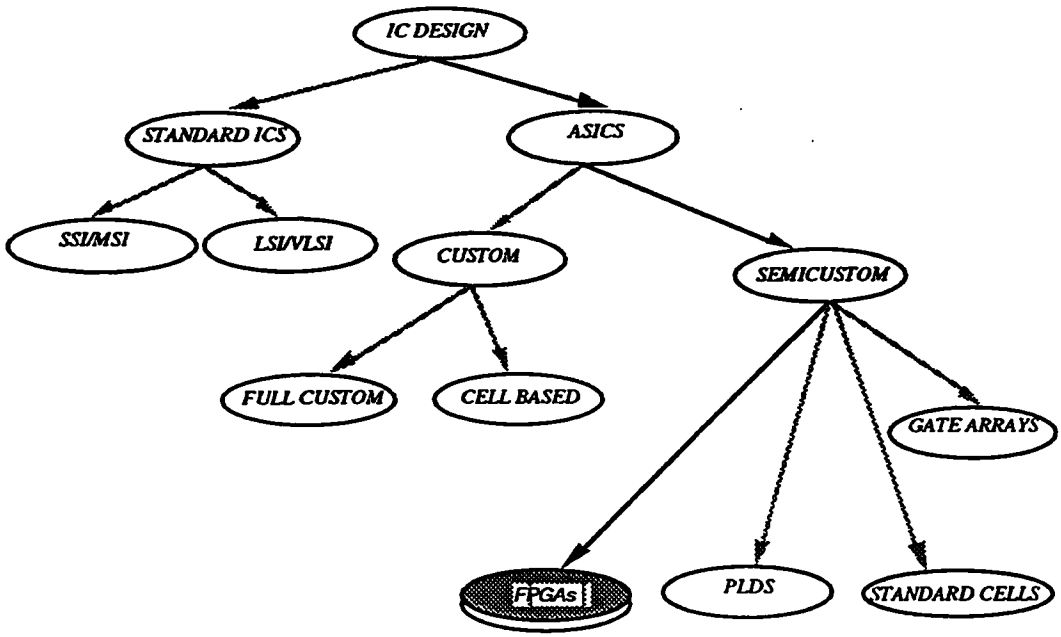


Figure 1.1: IC Design Alternatives

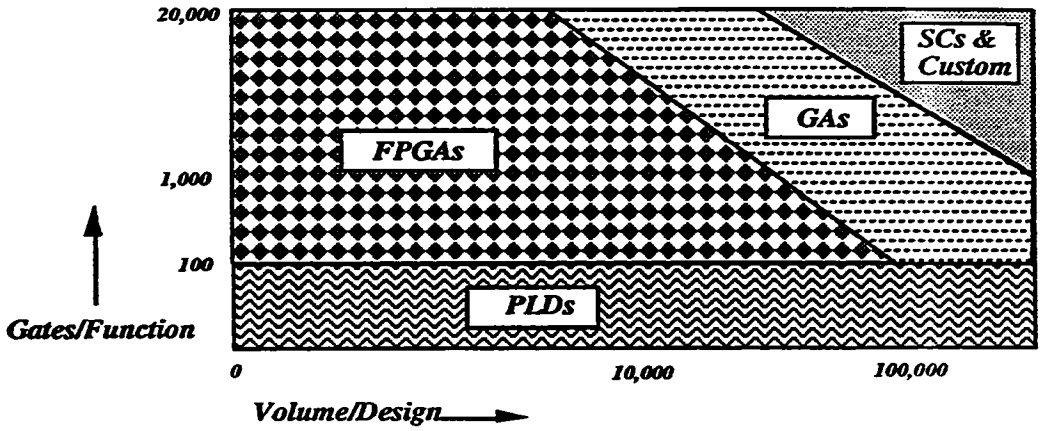


Figure 1.2: Density vs. Volume of Production for ASICs

To customise FPGAs to a given configuration, programmable switches (elements) are used. The programmable elements are built using two main technologies. The first uses an NMOS pass transistor whose gate is controlled by data stored in a static RAM cell [Car86]. In this case, programming refers to storing proper configuration data into the SRAM cell. The second uses anti-fuses [Ham88] to electrically connect or isolate specific nodes. An anti-fuse is a normally open electrical element which is shorted when stressed by a high voltage. In this case, programming refers to shorting the anti-fuse to connect two particular nodes or leave it open according to requirements. Programming an anti-fuse is an irreversible operation. Other programming technologies have also been reported, e.g. EPROM and EEPROM memory cells [Won89,Gup90].

In addition to the control and programming circuitry, FPGAs mainly consist of three basic building blocks: the programmable interconnect, the configurable logic modules and the configurable i/o blocks. The basic logic module of an FPGA provides the functional elements from which intended designs are constructed. Generally, it consists of a combinational part and a sequential part. Reported combinational parts include very

fine grain blocks (e.g., 2-input NAND gates [Has90]), medium grain blocks (e.g., multiplexer-based logic modules [Gam89]) and fairly large grain blocks (e.g., table lookup function generators [Car86]). Figure 3 shows an example of a large grain table lookup logic module. The lookup table is implemented as a general function generator which is customised into a particular function using a set of programmable elements (switches). The sequential part usually consists of one or more D-FFs with programmable control. The outputs of the logic module may be directly taken from combinational part or it may be registered through the D-FFs. They may also be provided with tri-state facility. Large fan-in AND/OR based logic modules have also been used, especially in the type of FPGAs derived from PLDs [Won89].

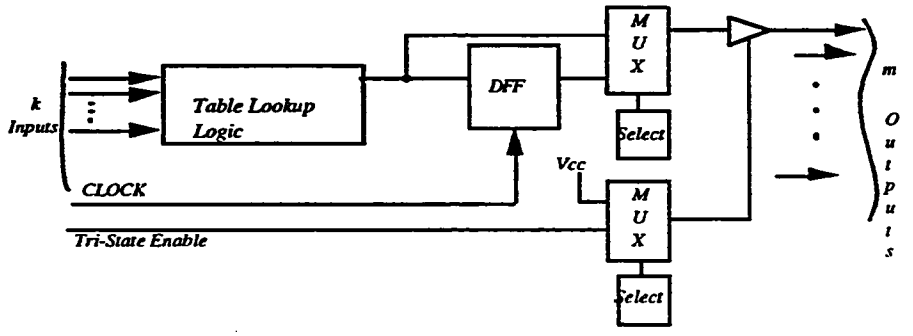


Figure 1.3: Table lookup Memory-Based Configurable Logic Module [Rose90b]

Figure 1.4 shows an example of a multiplexer-based logic module [Gam89]. In this case, Sequential functions are implemented by feeding back the module output to one of its inputs.

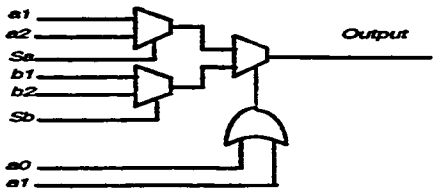


Figure 1.4: Multiplexer-Based Configurable Logic Module

The FPGA architectures have a programmable interconnect structure which provides routing paths to connect inputs and outputs of I/O and logic modules into logic networks. Interconnections between modules are composed of metal segments in one or two metal layers. Metal segments are connected using either anti-fuses or pass transistors controlled by SRAM, EPROM, or EEPROM memory cells. The interconnect segments may be regular in length and location [Car86] or they may have variable lengths [Gam89] and is generally distributed all over the array. Some architectures use a global interconnect bus as well as local interconnect lines running between logic modules [Won89].

The i/o blocks provide an interface between the external pins and the internal logic. Outputs may be direct or registered. An output may also control a tri-state buffer. Similarly inputs may be registered or direct. Figure 1.5 shows an example of a configurable i/o block.

Device programming refers to the process by which the user configures the FPGA programmable elements to implement the desired logic. Configuration data are usually shifted in serially to the array. Two shift registers are used, one defines the addresses of the elements to be programmed while the other defines their data. Figure 1.6 shows an example of such an arrangement.

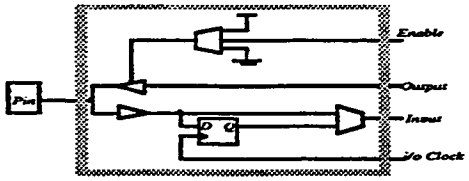


Figure 1.5: A configurable i/o block

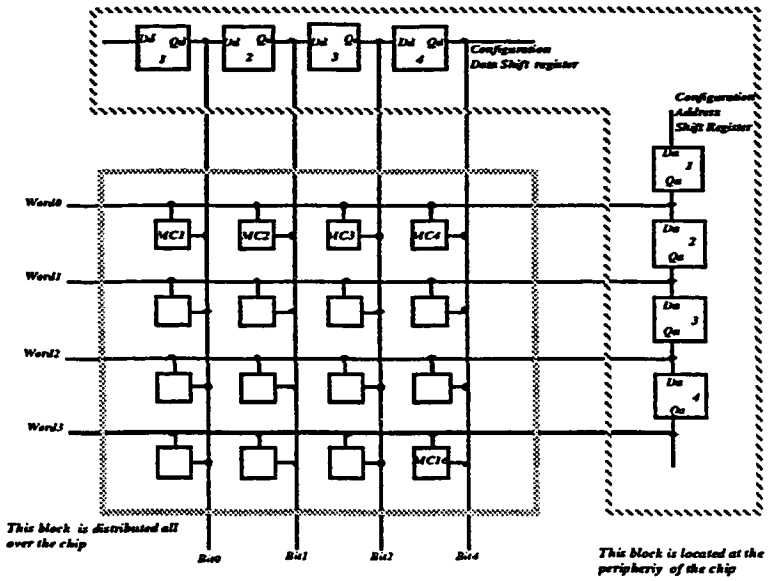


Figure 1.6: Configuration Memory Structure for Memory-Based FPGA

1.3. CAD Support

One of the advantages of an FPGA design approach is its short TAT compared to other design approaches. For FPGAs to retain this advantage, adequate CAD support is needed. For FPGA design style there are three principal phases: design entry, design implementation, and design verification.

1.3.1 Design Entry

Design entry refers to the process of specifying the logic of the circuit to be implemented within the FPGA. Schematic editors are the most widely used CAD tools for describing a new design. They rely on a macro library, and sometimes, they support a netlist interface. Boolean equations, truth tables, state machine descriptions, and even waveforms are among the many other forms of design entry used to specify the logic that the FPGA must implement.

Usually, most of the design entry forms come assisted with device specific CAD tools called fitters. The main task of these fitters is to optimize the logic to fit the available resources of the target FPGA architecture. Recently, hardware description languages (HDLs) have been suggested and used as a higher form of design entry. The logic of the circuit is described using some HDL. The HDL description is then synthesised to fit the target FPGA architecture. This approach is still in its infancy, and is the subject of intensive research.

Several FPGA and CAD tool suppliers are collaborating in an ad-hoc effort to define a standard language for the interchange of the logic of new FPGA designs. The language is called LPM (library of Parametrised Modules) and is for the descriptions of designs at the macro level. These descriptions may be generated manually or synthesised automatically. Each macro function can then be optimised to fit the target FPGA architecture.

1.3.2 Design Implementation

In most general terms, design implementation refers to the process of mapping the logic to be implemented into the configurable logic modules, the I/O blocks, and interconnect structure of a target FPGA architecture. The two principal tasks of this phase are placement and routing. Placement consists in assigning locations to the blocks while optimising a certain objective function and/or satisfying a number of criteria (e.g. geometric fit, routability, timing, etc.). Routing consists in assigning routing tracks to the signal nets so that all the blocks are properly interconnected.

For hierarchical design methodologies, partitioning tools are used to introduce one or many intermediate hierarchical levels. At a higher level, the description is in terms of major functional blocks. At a lower level, the description gives the details within each major block. At the lowest level, the logic of the circuit is given in terms of the configurable logic modules of the target FPGA architecture. Hierarchical design methodologies require a design library or design database, where all predefined macros are stored and available for instantiation.

The quality of the implementation CAD tools is measured by the quality of the produced FPGA design. Good tools should lead to efficient implementations with respect to area usage and most importantly with respect to performance (timing). Some work on performance driven physical design (mainly floorplanning, placement, and routing) has already been reported in the literature [Sutan91]. The techniques suggest coupling the timing verification step with the following physical design step. Besides verifying the design for logic related timing problems, the timing analyser also identifies the most timing critical paths and nets of the design. In some reported work, the timing analyser computes delay bounds/weights on all the nets of the design. Predictions from the timing analysis step are used by the placement/routing program to produce an implementation consistent with the initial user timing (speed) requirements.

Timing driven design methodologies require the accurate pre-characterisation of the timing properties of the configurable logic modules and I/O blocks of the target FPGA architecture. Such accurate timing characterisation requires that a good delay model, of the kind supported by the EDIF language, be adopted [You90a].

1.3.3 Design Verification

Design verification consists of checking the functionality as well as the timing of the proposed design. As opposed to other design styles (such as mask programmable gate arrays), the functional correctness of reprogrammable FPGAs is readily testable in the circuit implementation itself. Therefore, for such FPGAs only the timing properties of

the design need be checked. As mentioned in the previous subsection, the necessary timing attributes of the primitive elements of the FPGA (from which the configurable logic modules and I/O blocks are built) must already be known.

There are two general approaches used for timing verification: dynamic timing analysis, and static timing analysis. The dynamic approach performs a detailed functional level timing simulation of the design. The processing time of simulation is proportional to the number of events, which is exponential in the number of circuit inputs and circuit elements. Therefore, simulation has very high computer resource requirements in time as well as space. Furthermore, simulation can only report the existence of timing problems. It cannot be used to identify all of the critical paths or compute delay bounds on the interconnects for the purpose of performance driven physical design. Also, because it is unfeasible to simulate the design for all possible input test vectors, simulation may fail to detect some of the timing problems within the circuit.

With the increasing sizes of FPGA designs, static timing analysis is completely replacing dynamic analysis. Static timing analysis is based on a graph theoretical approach. The design is modelled as a directed graph with the blocks being the vertices, and the connections the edges of the graph. Unlike simulation, static timing analysis ignores the logic functions of the blocks. Therefore, this approach is less accurate than simulation and may point to some logically impossible paths as having timing problems. Proponents of this approach rightly claim, that it is safer to flag a false path as having timing problems than fail to report a path with actual timing problems. In a static analysis, the circuit is traced to enumerate all paths with long or short path

problems. The design is said to have long path problems if some of the signals have excessive propagation delays and violate the setup time requirement at the paths sinks. A design has short path problems when the delay on each of these paths is too short, which may violate the hold time requirements at he sinks of these paths.

A timing verifier using the static approach usually produces all of the timing critical paths and nets. It can also be easily augmented with the necessary routines to compute delay bounds or weights on the nets for the purpose of performance driven placement/routing [You90b].

1.4. Drawbacks of Existing FPGA Architecture

Anti-fuse based FPGAs have fairly flexible architectures and wiring interconnect schemes that reduce signal capacitance resulting in a reduced *ac* power consumption and signal delays. Anti-fuse programming, however, is a destructive irreversible process. Whereas this causes the configuration pattern of such FPGAs to be non-volatile, it also makes it non-reprogrammable. Being one time programmable, anti-fuse based FPGAs do not offer the flexibility required for new prototype designs. In addition, such FPGAs cannot be fully tested before shipment to customers since the anti-fuse devices cannot be properly tested without device destruction. SRAM-based FPGAs, however, are ideal for prototype designs. They can be easily reconfigured by storing different patterns in the configuration memory cells. Since they are fully reprogrammable, SRAM-based FPGAs can be fully tested before shipment to customers. One major disadvantage of such FPGAs, however, is the volatility of the

stored pattern where the FPGA configuration pattern has to be loaded each time the system is powered up. The configuration pattern can be stored in an EPROM memory or it can be down loaded from the processor. In addition to complicating the design, larger board area may be required to accommodate the configuration EPROM memory. Moreover, designs using such systems suffer from the lack of proper security where the configuration bits can be read and reverse engineered. Another disadvantage of SRAM-based FPGAs is the relatively large area of the programming element.

1.5 Scope of the Study

This work is an introductory study into the design of an FPGA architecture which is based on a novel[1] non-volatile static memory cell [Amin92]. The new design combines the user reprogrammability and full testability advantages of the SRAM-based designs together with the non-volatility and design security advantages of anti-fuse based FPGAs. The new FPGA architecture whose characteristics are given in Figure 1.7, relies on a new memory cell. The memory cell uses a charge pumped power supply that significantly reduces the programmable element resistance resulting in faster designs. The new cell, however, occupies a slightly larger area than the conventional SRAM cell. In addition, the new cell uses a flash EEPROM transistor, therefore requiring a more complex process.

[1] Patent Pending

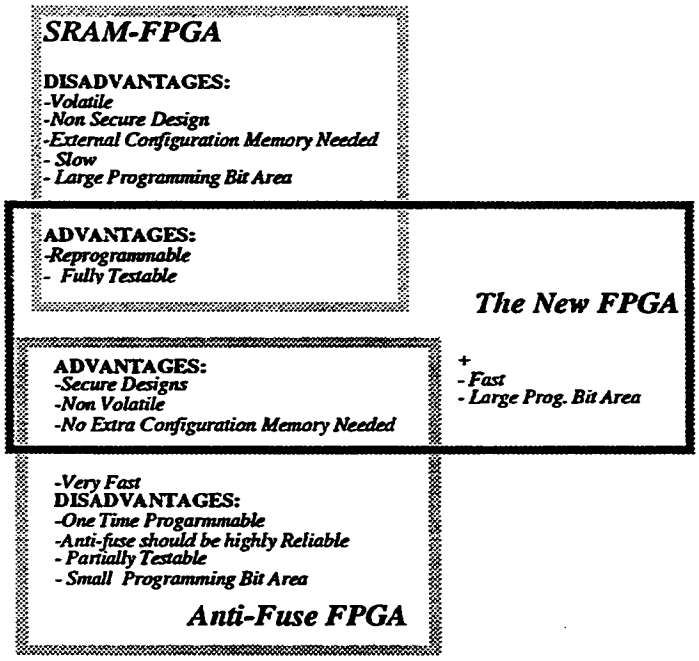


Figure 1.7: The FPGAArchitecture Characteristics

The study includes the choice of a proper logic module, an interconnect scheme and design of proper programming circuitry. The choice of the architectural parameters, e.g. logic block granularity and structure were guided by earlier empirical studies. The choice of the routing structure should be flexible enough to efficiently achieve full routability of a large number of circuits of a given size. In addition to the interconnect flexibility factors, choice of the necessary number of tracks and the required track lengths are critical parameters in this regard. A Stochastic model of interconnection of the FPGA circuits methods was employed to help decide the values of these parameters. These and other issues are adressed in the remainder of this thesis.

1.5 Organization:

This dissertation is organised in a bottom-up manner. In Chapter 2 we review the reported literature on major FPGAs where FPGA architectural parameters of interest to this work are stressed. In Chapter 3, the memory cell is described and and its programming modes are explained, and a detailed description of the design of the logic module is also given. In Chapter 4, the new FPGA interconnection structure and parameters are defined. Parameters critical to the FPGA design, e.g, the clock and power distribution systems are detailed in chapter 5. The overall floorplan of the FPGA is outlined in chapter 6. A summary of the results obtained from this work and suggestions for future work are given in chapter 7.

Chapter 2

Literature Review

Since their appearance in 1985, FPGAs have captured the interest of both the industry and academia. Reported FPGA architectures are targeted either for random logic applications or for special application, e.g. digital signal processing or high speed processor interfacing. In the following review, representative FPGA architectures are described. In addition, the major FPGA architectural parameters, e.g logic module granularity, interconnection parameters, and speed/area performances are discussed.

2.1 FPGA Architecture

2.1.1 SRAM Based FPGAs

The first FPGA family, the *L*ogic *C*ell *A*rray (LCA) [Car86, Xil91], was introduced in 1986. The generic architecture for the LCA is illustrated in Figure 2.1. The LCA is

based on a small logic module, the configurable logic block (CLB). Each block contains programmable combinational logic and a flip-flop capable of implementing any Boolean function of its input variables, as well as some sequential functions. The blocks are distributed over the core of the chip in the form of a matrix. At the periphery of the chip programmable input/output blocks are used. Each such block can be independently programmed as an input buffer (TTL or CMOS compatible), as an output buffer with tri-state, or as a bidirectional input/output buffer. In addition, each input or output block contains flip-flops to allow for latched input or output options. The interconnect resources include a 2 layer metal network of lines which run in routing channels horizontally, and vertically between rows and columns of the CLBs. Typical interconnect lines are shown in Figures 2.2 and 2.3. Programmable switches connect the inputs and outputs of CLBs and I /O blocks to nearby metal lines. At the intersection of columns and rows, cross point programmable switches are provided to switch signals from one path to another. A signal may go through several pass transistors before reaching its destination. Long lines running the entire length and width of the chip, bypassing interchanges, are provided for the distribution of critical signals and the system clock with minimum delay and skew. Direct interconnect lines are the most efficient way to connect adjacent CLBs or I/O blocks. Signals routed this way undergo the least amount of delay. The array has an SRAM-based programming technology, with the programming data shifted serially into the device. The memory array does not have a conventional memory system structure, rather it is distributed throughout the whole core of the chip. The combinational logic section of the CLB is implemented as a lookup table. This scheme allows a constant delay through the CLB

regardless of the combinational function implemented. Each CLB has four general purpose inputs in addition to a clock input. Two outputs are available from each CLB. These outputs can be used to drive the interconnect or they can be fed back internally to the input of the CLB.

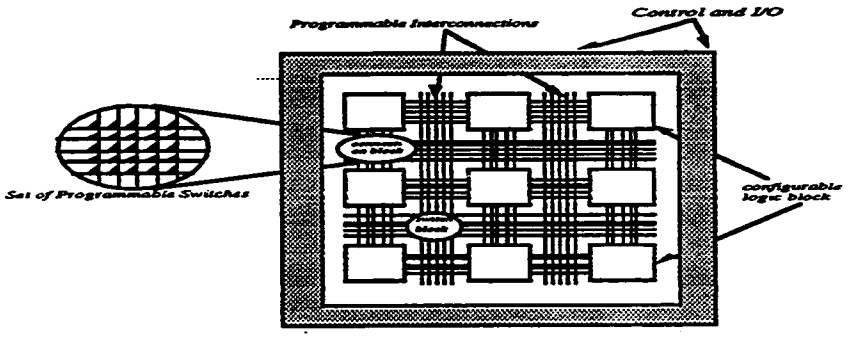


Figure 2.1: Generic LCA Architecture

The second generation of the LCA, [Hsi87, Hsi88] differs from the first generation in density, CLB structure, i/o blocks, and an added internal bus structure that can be used to route multiplexed signals. The number of flip-flops per CLB has been increased to two and the number of CLB inputs (data and control) has been increased as well. Five of these input lines are combinational inputs while the remaining five are used for flip-flop control.

A third generation of the LCA family was introduced in 1990 [Hsi90, Xil91a]. The CLB has a total of thirteen inputs and four outputs with three independent function generators. Two generators, with four inputs each, are provided with two flip-flops while the third has three inputs and the same number of flip-flops. The architecture

allows the 32 configuration memory cells of the 3 function generators to be configured as a read/write memory. It can be configured as a 16x2 RAM or as a single 32x1 RAM. The inputs to the configuration memory constitute the address lines with one or two data lines. Other added features of this family is the possibility of implementing decoders and fast adders. Implementing wide decoders in previous generations required a large number of CLBs. The decoders were not fast enough for certain applications. Dedicated hardware was also added to each CLB to implement fast adders where an arithmetic carry is generated and can be used by the next CLB. To simplify board level testing of electronic assemblies in which the LCA will be integrated, this family implements the IEEE1149.1 Boundary Scan standard.

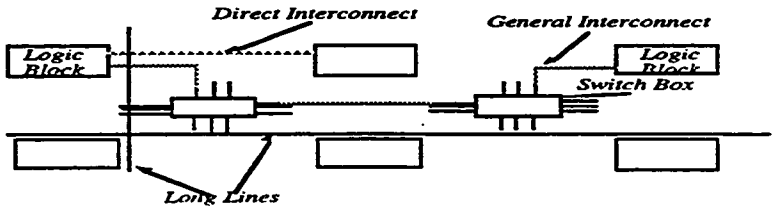


Figure 2.2: Interconnect resources for the LCA

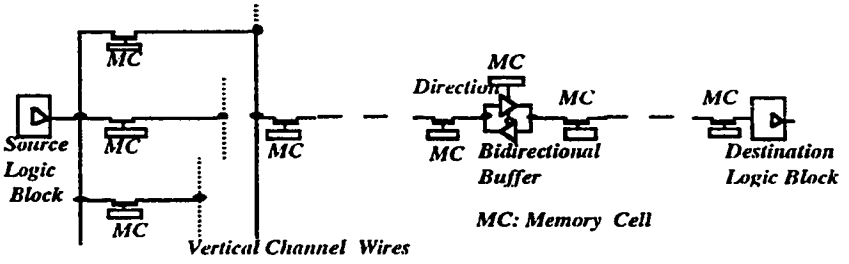


Figure 2.3: Typical Simple Signal Path for Memory-Based FPGA

2.1.2 Anti-Fuse Based FPGAs

In 1988, an anti-fuse-based [Ham88] family of user configurable gate arrays [Ayat88, Gam89] was announced. Its generic architecture is given in Figure 2.4. The interconnect structure is similar to that of a conventional row based gate arrays with the exception that the channel wire segments are predefined not only in number but also in length. Wire segments of variable lengths are used. Each input and output of a logic module is connected to a dedicated vertical wire segment. Such input/output wire spans several horizontal channels in case of an output signal, and one channel in case of an input signal. Some other vertical segments are used as feedthrough logic modules to allow signal flow between channels. Programming of the interconnect anti-fuses is done using a number of pass transistors and control logic at the periphery. The programming data are shifted serially into the device. A logic module consisting of three 2-input multiplexers and a 2-input OR gate is used. All modules are single output gates. The logic module implements all combinational functions of 2 input variables, many of 3 or 4 variables and others ranging up to 8 variables. The sequential functions are configured using one or more modules with appropriate feedback routing. The architecture uses long lines to avoid using resistive anti-fuses along critical signal path.
A higher integration second generation of this family of devices was announced with 8000-16000 gates [Ahr90]. The architecture uses a mix of two logic modules: one optimised to accommodate high fan-in combinational basic functions such as wide AND gates. The second is optimised for configuring basic sequential functions such as latches and flip-flops in addition to many combinational basic functions of up to seven

inputs.

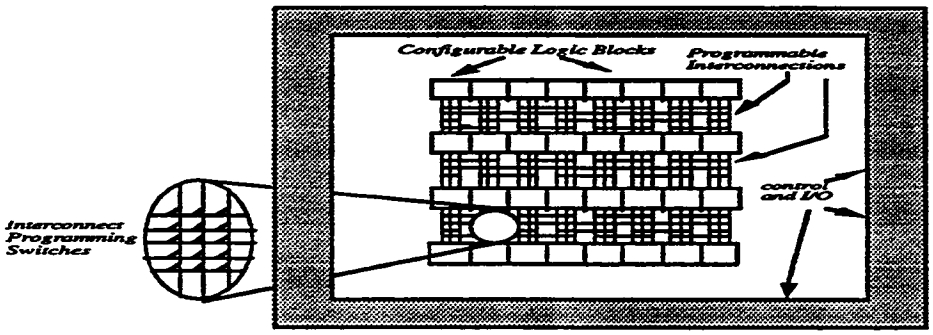


Figure 2.4: Anti-fuse-Based FPGA Generic Architecture

2.1.3 PLA-Based FPGAs

Another FPGA architecture [Wong89], directly derived from EPLD uses EPROM transistors as programming elements, in addition to multiple AND/OR arrays as shown in Figure 2.5. By using multiple AND/OR arrays, each with a fixed number of inputs, this architecture avoids the conventional PLD exponential growth in area with the number of inputs. This new architecture uses an interconnect scheme which connects multiple distributed logic modules through buses. The integration of 5,000 gates was reached by such architecture. Each logic array block consists of a number of macrocells (basic logic modules). The programmable interconnect array is a separate array with a small number of lines driving the inputs of the logic array blocks. The programmable interconnect array global busses consist of outputs of all macrocells and all i/o pins,

which are decoupled from the macrocells. This approach provides global connectivity throughout the chip and limits the size of each logic array block since it routes only the needed signals.

2.2.4 Other FPGA Architectures

Several other architectures have been reported. A RAM-based reconfigurable FPGA (called Labyrinth) uses a basic cell with four inputs and four outputs. The basic cell can be configured as a single bit register, a half adder, an identity function (wire) or as the logical constant "1" [Fur90]. This architecture has been used to map algorithms directly into hardware, e.g., video compression algorithms used in multimedia.

Another SRAM-based architecture [Kow90] uses content addressable memories to implement logic functions.

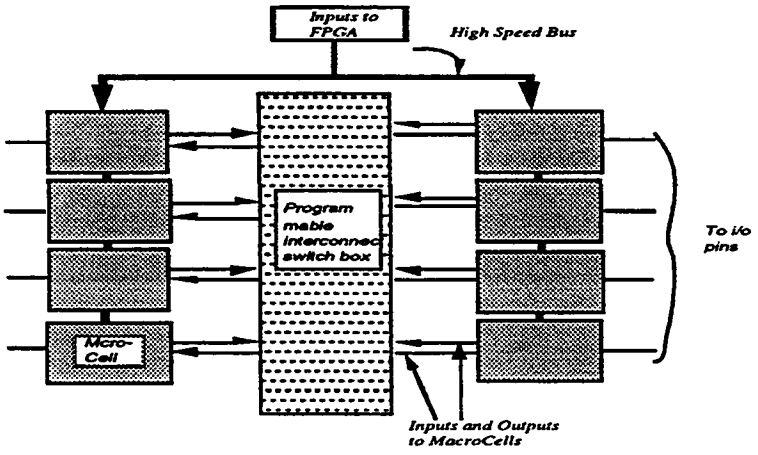


Figure 2.5: AND/OR Macrocell-Based FPGA Architecture

The architecture uses an efficient logic module interconnect structure to reduce the delay in signal propagation delay. The logic module contains a combinational logic function generator and two flip-flops. It has four general purpose inputs, a flip-flop load, together with control inputs and four outputs. A dynamically reconfigurable array based on Write Only Memory (WOM) has also been reported [Has90]. It consists of a conventionally structured static RAM to control the connections in addition to an array of NAND gates in the form of a sea of gates. Due to the speed at which programming data can be shifted in this device, it can be used to implement hardware subroutines.

2.2 FPGA Architectural Parameters

Several theoretical studies were performed to relate the various FPGA design parameters to its area and performance. Issues such as logic module granularity, logic module architecture and the interconnect structural parameters that are of concern to the FPGA architecture have been investigated.

2.2.1 Logic module Granularity

Logic module granularity refers to the size of the logic module and hence its functionality. The logic module functionality and area, i.e. granularity, is generally related to its number of inputs. This is particularly true for lookup table logic modules. Thus, the number of inputs to the logic module is considered as a good measure of its granularity [Rose90b].

The relationship between the logic module granularity and the total area required to implement a given circuit was empirically investigated [Rose90b].In performing this study, several industrial circuits were mapped onto FPGA architecture similar to LCA architecture. Technology mapping of each circuit into an FPGA with this architecture is performed and the mapped circuit total area is calculated. A general logic module with one output and a varying number of inputs was used. This was performed for five different technologies using logic modules with and without D flip-flops while varying the number of logic module inputs from two to ten. The FPGA was modelled as an NxN array with horizontal and vertical routing channels. Based on these models, the total area required to implement a given circuit was estimated as a function of the number of logic module inputs and programming technology. In this study two problems were addressed. The first is the choice of the number of logic module inputs required to minimise the total area independent of the type of programming technology. The second is the effect of including D flip-flops in the logic module on the total required area as shown in Figure 2.6.

Circuits in standard cell array design style and programmable array logic (PAL) methodology were also mapped into FPGAs [Rose90b]. FPGAs that use table lookup logic modules with a varying number of inputs and programming technologies were also used. A simple model for each logic module and its associated routing area was employed (Figure 2.6). The result of the experiment showed that the minimum total area varies slightly with the programming technology. In addition, it was found that the

average number of logic module inputs that minimises the total area is between 3 to 4. The total FPGA area was found to be largely dominated by the routing area which represents from 50% to 90% of the total area depending on the logic module granularity. The inclusion of a D-FF was found to reduce the total average area. In some implementations, the ratio of area without D-FF to the area with D-FF was 2.8 while the area of the D-FF itself contributed only 25% of total area. This was attributed to the fact that implementing D flip-flops using logic modules requires more area.

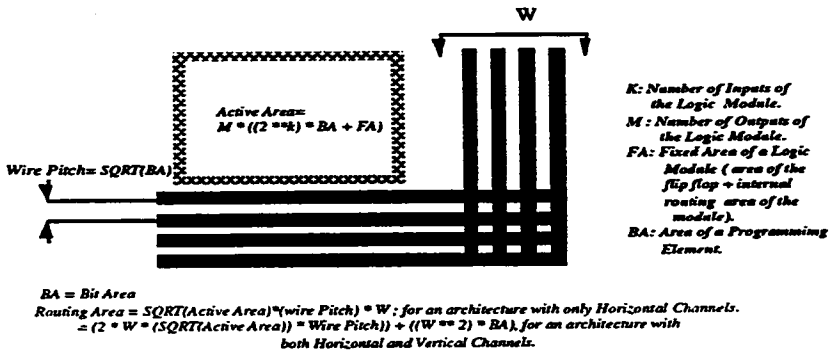


Figure 2.6: Area Model For Table Lookup FPGAs

Another experimental study was conducted to investigate antifuse-based FPGA architectural parameters[Gam91]. It explored tradeoffs between various architectural parameters affecting the area and performance of the FPGA. In particular the effect of the basic cell granularity on the performance and the area of the FPGA. The FPGA model used in this study was similar to that of Figure 2.4. The basic logic module used consists of a combinational table lookup circuit of k inputs and m outputs with an

optional flip-flop at the output. The experiment showed the following:

- The total number of cells in a design decreases as the cell granularity increases.
- The average net delay is an increasing function of the number of inputs of the logic module.
- The critical path delay in a design varies with the number of logic module inputs as well as the switching delay of the programming element (SRAM or anti-fuse) time constant t_s.
- Technologies with a small programming element time constant t_s favour small module granularity, while those with larger time constant favour larger granularity.
- Adding a second output can significantly reduce the total number of cells required to implement a design. Further increase in the number of outputs does not lead to any further gain in area.

2.2.2 Logic Module Architecture:

The effect of logic module architecture on the speed of the FPGAs was investigated [Sat92]. Four classes of logic module architectures have been explored. These are NAND gates, multiplexer configurations, lookup tables and wide AND/OR gates. An experimental approach was used to measure the speed of the FPGA for each of these logic module architectures. A major limitation of the FPGA overall speed was found to be the RC delay of the programmable element. This fact causes the mask programmable

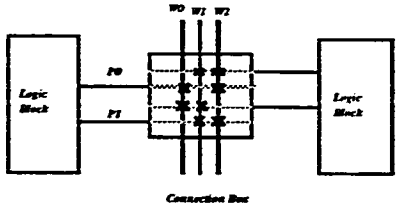
gate arrays (MPGAs) to be much faster than FPGAs since signal delays are proportional to the number of programmable switches in their paths. The performance of FPGAs can be improved by reducing the number of programmable elements used in routing critical paths. One way to overcome this problem is by using logic modules with higher functionality. This, generally, reduces the number of logic module levels in the critical path. The experiment also showed that five- and six-input lookup table modules exhibit the lowest total delay. The multiplexer-based module comes close behind. The module based on NAND gates exhibited the largest delay, while the delay of the wide AND/OR gates was between those two limits. The dominant part of the total delay was shown to be mostly due to the interconnect delay.

2.2.3 Interconnect Structural Parameters

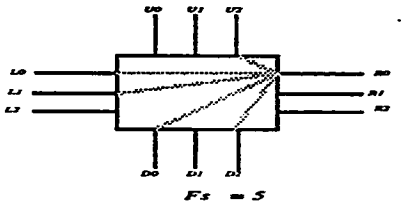
The effect of flexibility of the interconnect structure of FPGAs on routability and on the wiring resource requirements was studied [Gam91, Rose91]. The FPGA model adopted in this study is similar to the LCA architecture. Flexibility was defined as the total number of choices offered to each wire entering an interconnection block as illustrated in Figure 2.7. It was shown that, for high routability the connection box should be highly flexible, while the switch blocks are not required to have such high flexibility. In addition, a trade off between the switch and connection blocks up to a certain value of the switch flexibility was observed.

The study showed that there is minimum flexibility beyond which it is easy to achieve

near minimum possible number of tracks. Using a number of industrial circuits, the study showed that the minimum number of switches occurred with switch flexibility Fs between 3 and 4 and a connection block flexibility to the number of tracks per channel ratio (Fc/W) between 0.7 and 0.9.



Connection Box flexibility: Fc = 2
W1, W2, W3 are the vertical channel tracks
P1 and P2 are the primary i/o of the logic module entering the box.



Switch Box Flexibility:Fs=5
L0-L2, R0-R2 and U0-U2 and D0-D2 are the horizontal and vertical channel segments. respectively.

Figure 2.7: Connection Box and Switch Box Flexibilities

CHAPTER 3

THE NVRAM PROGRAMMING TECHNOLOGY AND THE UPLM

3.1 The Programming Technologies

The programming technology for FPGAs is the methodology through which the interconnect, the logic module and the input/output blocks are configured to build the user logic. These programming technologies can be classified in three categories as shown in Figure 3.1. These categories are antifuse FPGAs, floating gate FPGAs and SRAM based FPGAs.

The hard configurable scheme is used in anti-fuse based FPGAs. In this scheme, the interconnect switches and the logic module configuration switches have small area overhead due to the small size of the programming element. The anti-fuse has small parasitic resistance and capacitance. This reduces the signal transition time and propagation delay. Antifuse FPGAs suffer the disadvantage of being one time programmable. The switch programming being an irreversible process prevents full testability of these switches since a switch can not be tested for connection unless it is irreversibly programmed.

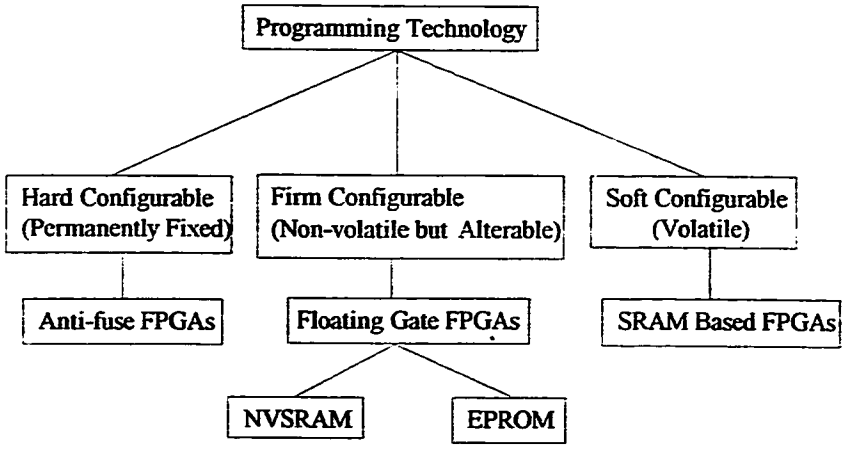


Figure 3.1: Classification of FPGAs According to Their Programming Technologies

The soft configurable scheme uses SRAM cells for the configuration of the device. An area overhead is required to store the state of the interconnect switches, the logic module and the i/o blocks configuration bits. It suffers also the disadvantage of being volatile requiring an external loading device. The advantage of the soft reconfigurable scheme lies in their flexibility to be reconfigured as many times as wanted, in addition to being fully testable.

The firm configurable scheme is represented by both the EPROM and NVRAM technologies. The EPROMs technology needs ceramic chip carrier to host the quartz window to allow UV light reach the chip for its erasure. The technology suffers from the low Write/Erase endurance and long programming time. The NVRAM scheme is an alternative combining the advantages of both the hard, and soft configuration schemes.

3.1 The New Static NVRAM Cell

The new static NVRAM[1] memory cell has been used in this work. The cell uses floating gate FEEPROM [Amin92a]. The logic diagram of the cell is given in Figure 3.2

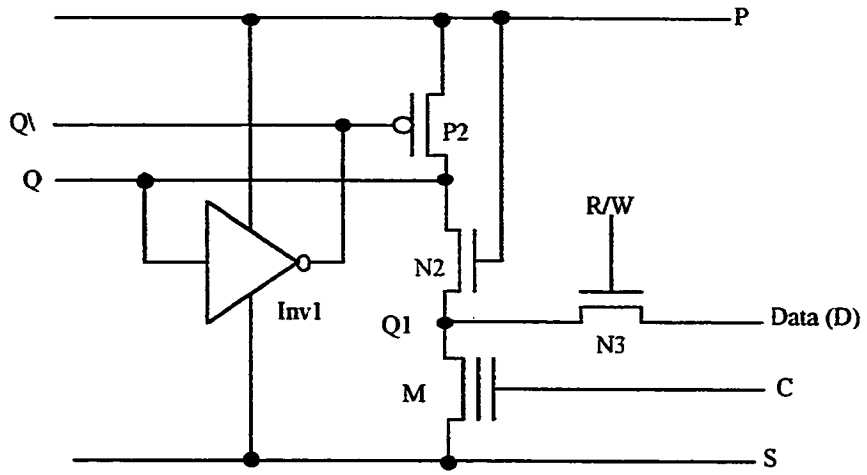


Figure 3.2: The NVRAM Cell

The cell power lines (P, S) as well as the control lines C, R/W and Data assume different values of voltages depending on the operating mode of the cell. The memory cell has five modes of operation: a normal mode, a read mode, a programming mode, an erase mode, and an initialization mode.

The full description of the memory cell can be found in [Amin92a, Amin92b]. In the following a brief description of the memory cell operating modes is given.

[1]Patent Pending

3.2 Modes of Operations of the New Memory Cell

3.2.1 Normal Mode

In this mode, the configuration memory cell provides the configuration control signals (Q and Q\) required to configure the interconnect, the logic modules and the i/o blocks of the FPGA into a particular design. In this mode, P is charge pumped to $Vcc'=Vcc + 2V_{tn}$. Vcc and V_{tn} are the supply and threshold voltages respectively. This boosts voltage at the gates of the interconnect pass transistors, considerably reducing their drain resistances. This allows full rail to rail signal routing eliminating the threshold voltage drop suffered by earlier SRAM-based FPGAs.

3.2.2 Read Mode

This mode of operation allows reading the stored configuration pattern. It can be used for the verification of the FPGA programming.

3.2.3 Programming Mode

This mode is equivalent to a nonvolatile write operation of logic 1 data into the memory cell. This is accomplished by programming M into the high threshold voltage state. The programming is performed in three steps

1. *Voltage setup:*

- $V_C = 0$ (OFF floating gate "M"),
- $V_R = V_{PP}'$ (ON access transistor "N3"),
- $V_S = V_{PP} = 12$ V (S-line acts as the programming power supply line),

- $V_D = V_{PP}$ for cells that should maintain their previous state, while VD = 0 for cells that should be programmed,
- $V_P = V_{CC} - V_{tn}$ (no need for charge pump P during this mode),

2. *Programming step*;

A programming voltage V_{PP}' pulse (14 volts) is applied to V_C, thus only programming all floating gate transistors whose data line D is grounded.

3.*Reset step*:

All high voltages are reset (note that since $V_C = 0$ V no discharging takes place through M thus avoiding spurious programming), hence,

- $V_S = 0$,
- $V_R = V_{CC}$,
- $V_D = 0$.

3.2.4 Erase Mode

This is equivalent to a non-volatile write operation of logic 0 data into the memory cell. This is accomplished by erasing the floating gate transistor M to its low threshold state. Either the source or the drain of the transistor M may be used as erase electrode.

3.2.5. Initialization Mode

For proper operation of the FPGAs, all the configuration memory cells should assume their proper states after power-up. Thus, all configuration control signals (Q and Q') must assume their correct values as implied by the state of floating gate transistors of their

respective memory cells. This operation is performed by the proper transistor sizing of the memory cell.

A summary of the static NVRAM cell typical operating conditions is given in table 3.1. The memory cells will be distributed throughout the array with cells in the same row sharing the R/W and P control signal. Cells in the same column will share the Data line, the S line and the C control line.

In the following paragraphs a full description of the logic module will be given and how this memory cell have been used in the design. In the following chapter, the memory cell will be used to design the FPGA interconnect structure.

Table 3.1: Modes of Operation of the Static NVRAM Cell

Voltages / Modes	Vp	Vs	Vc	Vr	Vd
Normal	Vcc'	Gnd	Vcc	Gnd	X
Read	Vcc'	Gnd		Vref	Vcc
Programming					
1) Voltage Setup	Vcc-Vtn	Vpp	Gnd	Vpp'	Vpp,Gnd (*)
2) Programming	(**)	(**)	Vpp'	(**)	(**)
3)Reset	X	Gnd	Gnd	Vcc	Gnd
Erase	Vcc-Vtn	Float	Gnd	Vee'	Erase Pulse
Initialize (***)					

(*) V_{PP} is taken for cell that have to keep their previous states, and Gnd for cells that have to be programmed.

(**) Previous values of the voltages are kept

(***) Conditions have to be set by the cell design parameters (e.g, transistor sizes).

$V_{PP} = 12V$, $V_{EE}' = 16V$, $V_{CC}' = V_{CC} + 2V_{tn}$

3.3 The Logic Module Architecture and Granularity

3.3.1 The Logic Module Architecture

The logic module provides the basic functional elements from which the user's logic is constructed. The basic logic module architecture of the FPGA affects the speed at which the implemented circuit can operate. Four classes of basic logic modules of the FPGAs were reported [Has90, Car86, Gam89, Won89]: NAND gates, table lookup based, multiplexer configuration and wide AND/OR plane based logic modules. These logic modules are shown in Figure 3.3. The NAND gates based logic module uses 2 inputs NAND gate as basic logic module. Such a fine grain module is used for both logic and routing [Has90]. In an table lookup based logic module of n inputs, a memory of 2^n is used to store the truth table of the function. The multiplexer is a universal function generator. It generates combinational functions by connecting its inputs to either 0, 1 or to signals from the interconnect line segments. The wire AND/OR logic modules are used in PLA-based FPGAs, and are very useful for implementing wide fanin functions.

One important factor which affects the FPGA performance is the granularity of the logic module. This factor affects both the speed of operation and the total area required to implement designs. In addition to the logic module granularity, speed is also affected by the interconnect structure For FPGAs the interconnect delay is a major component of the overall delay. This is due to the fact that the FPGA interconnect contains significant resistance and capacitance incurred by the programmable switches present in the signal paths. FPGA speed can be increased by reducing the number of programmable switches in the critical signal paths. Speed can thus be enhanced by using logic blocks with high functionality such that the number of logic module levels, and hence the number of switches in the critical paths is minimized

Figures 3.4 shows an example of a four input AND function implementation using 2-input AND gates as compared to its implementation using wide AND gate of 5 inputs.

In the first case, the function is implemented in one level of logic (Figure 3.6a) while in the second case it is implemented in three logic levels (Figure 3.6b). In the first case the switches are in parallel and incur one switch delay, whereas in the second case the switches are in series and thus incur 3 switch delays. However, the wide AND gates have larger combinational delay compared to the 2-input AND gates.

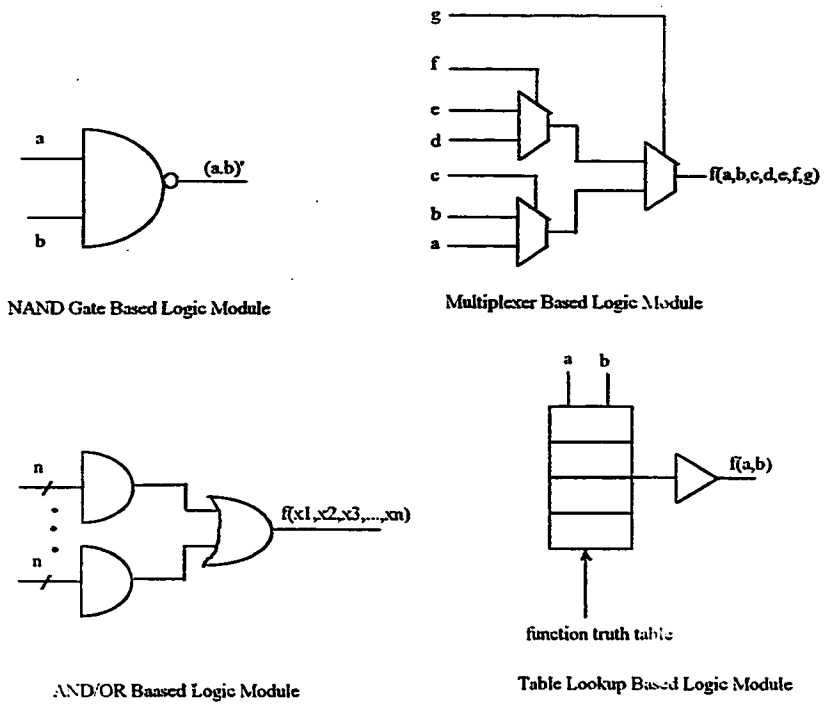


Figure 3.3: FPGA Logic Modules

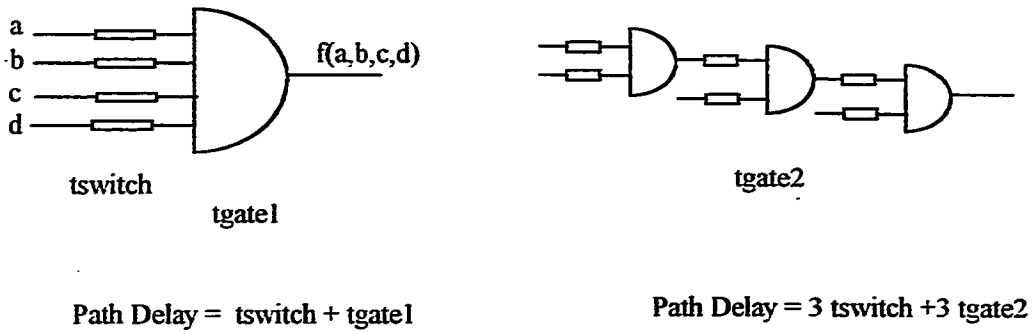


Figure 3.4: Module granularity vs. speed

This increased combinational delay is justifiable if it is less than the reduction in routing delay.

Considering the four reported logic block types, it has been shown that the wide AND/OR based logic modules have a larger combinational delay [Sin92] as compared to the three other logic blocks, while the 2-input NAND gate based logic module gives the largest total average delay. The table lookup based logic modules exhibits the lowest delay. The multiplexer based logic modules come close behind the lookup table based logic module. *Since table lookup modules exhibit the least average delay they have been adopted in this work.*

Furthermore, the inclusion of a D flip-flop in the logic module was shown to reduce the chip area [Rose 90b], since more area is required if they are implemented using purely combinational logic blocks. D flip-flops are needed in most circuits.

3.3.2 Logic Module Granularity

The number of outputs of the logic module also affects the total area to implement a circuit. It has been shown that the number of modules to implement a circuit in an FPGA

is reduced if the number of outputs is increased [Gam91]. This reduction is significant if the number of outputs is increased from one to 2. No substantial reduction is noticed if the number of outputs is increased to more than two outputs.

3.3.3 Table Lookup Logic Module

The circuits proposed by Yau and Tang [Yau70] can be used to implement any random Boolean function. These circuits and those having similar properties are known as the universal logic modules (ULMs). Modules capable of realizing any combinational functions of n-or less input , are called n-ULM. The number of combinational functions that can be realized by an n-UPLM is 2^{2^n}.

Any function $f(x_1, x_2, x_3, ..., x_n)$ of n variables can be expanded with respect to any r variables $r \leq n$ in the following form:

$\forall\ r \leq n$

$$f(x_1, x_2, x_3, ..., x_n) = \sum_{i1,i2,...ir} y_1 y_2 .. y_{r-1} y_r f(i_1, i_2, i_3, ..., i_r, x_{r+1}, ..., x_n) \qquad (3.1)$$

where, $i_k \in \{0, 1\}$, and

$$y_k = x_k, \text{ if } i_k = 1$$
$$= \bar{x}_k, \text{ if } i_k = 0.$$

For example the function $f = f(x_1, x_2, x_3, x_4, x_5)$ may be expanded with respect to the variables x_1 and x_2 as follows:

$$f(x_1, x_2, x_3, x_4, x_5) = \bar{x}_1 \bar{x}_2 f(0, 0, x_3, x_4, x_5) + \bar{x}_1 x_2 f(0, 1, x_3, x_4, x_5) + x_1 \bar{x}_2 f(1, 0, x_3, x_4, x_5) + x_1 x_2 f(1, 1, x_3, x_4, x_5)$$

The expansion of a function of four variables x_1, x_2, x_3, x_4 with respect to all of them will be as follows:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) = {} & \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 f(0,0,0,0) + \bar{x}_1\bar{x}_2\bar{x}_3 x_4 f(0,0,0,1) \\ & + \bar{x}_1\bar{x}_2 x_3\bar{x}_4 f(0,0,1,0) + \bar{x}_1\bar{x}_2 x_3 x_4 f(0,0,1,1) \\ & + \bar{x}_1 x_2\bar{x}_3\bar{x}_4 f(0,1,0,0) + \bar{x}_1 x_2\bar{x}_3 x_4 f(0,1,0,1) \\ & + \bar{x}_1 x_2 x_3\bar{x}_4 f(0,1,1,0) + \bar{x}_1 x_2 x_3 x_4 f(0,1,1,1) \\ & + x_1\bar{x}_2\bar{x}_3\bar{x}_4 f(1,0,0,0) + x_1\bar{x}_2\bar{x}_3 x_4 f(1,0,0,1) \\ & + x_1\bar{x}_2 x_3\bar{x}_4 f(1,0,1,0) + x_1\bar{x}_2 x_3 x_4 f(1,0,1,1) \\ & + x_1 x_2\bar{x}_3\bar{x}_4 f(1,1,0,0) + x_1 x_2\bar{x}_3 x_4 f(1,1,0,1) \\ & + x_1 x_2 x_3\bar{x}_4 f(1,1,1,0) + x_1 x_2 x_3 x_4 f(1,1,1,1) \end{aligned}$$

The general logic diagram which generates these functions is shown in Figure 3.5 The circuit performing this function is called function generator.

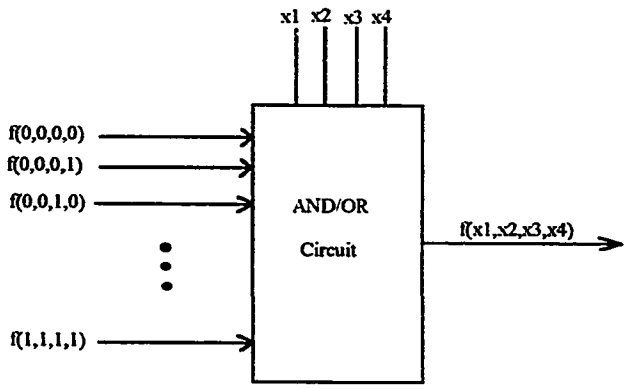


Figure 3.5: General Function Generator

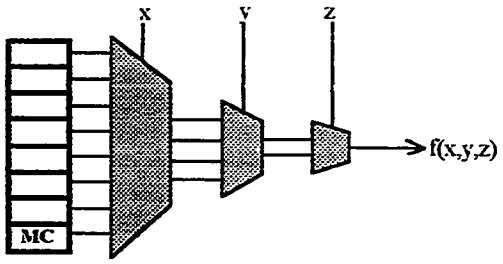
3.3.4 Residue Functions and the UPLMs

The primary inputs to the function generator (ULM) (e.g,x_1, x_2, x_3, x_4 in the previous example) are called the side terminals, while the terms f(i_1, i_2, i_3, i_4) are called the front terminals or the residue functions. The residue functions when fully expanded (r= n) have constant values (either 0 or 1) representing the truth table of the implemented function f. Thus, the function can be implemented as a multiplexer with its inputs fixed to either Vcc or ground representing the residue functions while the side terminals are used as select inputs. Antifuse FPGAs logic modules use multiplexers whose inputs are not tied to 0's or 1's but rather to input variables. In that case the residue functions are not fully expanded with respect to n ($r < n$). The residue functions can also be stored as a table in alterable memory cells (table lookup). *This is the implementation used in SRAM based FPGAs and is adopted in this work.* Since the adopted table-lookup module uses a table of alterable/programmable memory cells it will be referred to as a Universal Programmable Logic Module (UPLM) is obtained. The UPLM can implement any Boolean function of its n side terminals using the same physical configuration by proper choice of the table contents according to the function to be implemented.

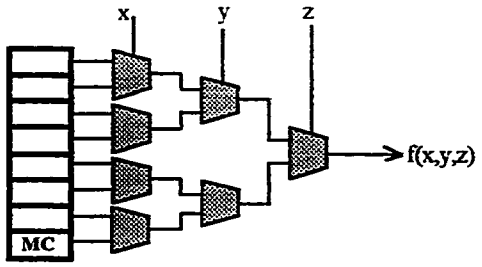
Equation 3.1 describes the function generator acting as a 1 of 2^r selector where the 2^r data lines are the residue functions and the r control variables are the inputs of the function. The output of the selector is the desired function to be implemented by the function generator.

There are several possible ways to implement function generators some of which are depicted in Figure 3.6 for the case n=r=3. In Figure 3.6a and 3.6b , the logic module is based on the selector circuits (regular and non regular implementations) whereas in Figure 3.5c it can be thought of as a conventional memory organization with only one bit-wide

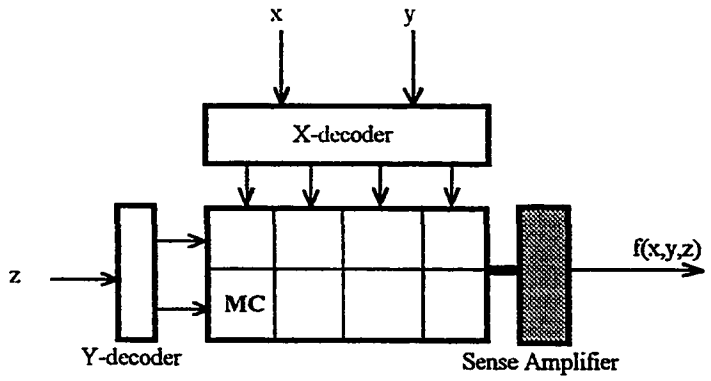
data bus. The address bus consists of the inputs of the function. Even though that scheme is simple, its implementation is quite involved due to the need for decoders and sense amplifiers.



a. General MUX Implementation 1



b. Modular MUX Implementation 2



c. Regular Memory Implementation

MC: memory Cell

Figure 3.6: Implementations of Table Lookup UPLMS

The implementation of the function generator using AND/OR CMOS technology with pull-up and pull-down networks requires a larger number of transistors as compared, the implementation of such generator using NMOS pass transistor logic [Whit82]. This implementation, significantly reduces the power consumption and the delay through the network.

3.4 The UPLM Design

Fine-grained FPGA logic modules have the advantage of high logic utilization, while coarse-grained ones enjoy higher performance. Thus, it is desirable to choose the granularity of the UPLM two main criteria are optimized:

1) *Area efficiency*:

An area efficient logic module has a high functionality per input/output pin. In addition, the architectural choice of the module granularity depends on the used programming technology. Thus, the number of inputs of the logic module should minimize the total average area of the implemented logic. This has been shown to fall between 3 and 4 for the SRAM and anti-fuse programming technologies[Rose 90b,Gam 91].

2) *performance:*

In addition to the area efficiency, the number of inputs of the logic module also affects both the overall speed and power consumption of the FPGA. In the selector implementation, pass transistor logic (Figure 3.6) is used. Typically, the voltage across the transistor is very small, and the current drawn from the supply is also small, resulting in a small power consumption. Figure 3.7 shows a simplified model of the pass transistor.

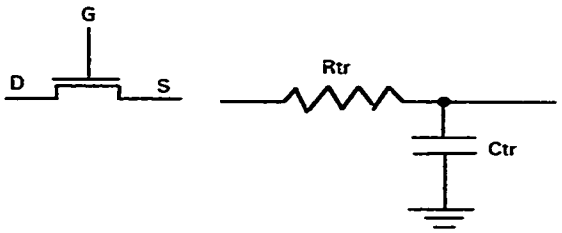


Figure 3.7: Pass Transistor and its Simplified Equivalent Circuit

The time constant of this pass transistor model is given by:

$$\tau_{tr} = C_{tr}R_{tr}$$

Where C_{tr} and R_{tr} are the gate capacitance and on-resistance of the pass transistor, respectively. This time constant gives a good estimate for both the high-to-low and the low-to-high propagation delays through the pass transistor. It represents the time to charge t_r to 63% of its final value and to discharge it to 78% of its initial value [Hor83]. The transistor on resistance is given by:

$$R_{tr} \approx (L/W)/(\mu C_{gox} V_D)$$

Pass transistor logic function generators use several cascaded transistors. The delay in each enabled path of n-cascaded pass transistors can be approximated by [Hor83]:

$$\tau eff = (n(n+1)/2)RtrCtr$$

It can be seen from this formula that the total delay through a series of pass transistors varies quadratically with the number of pass transistors in the path. Such delay is generally independent of the function implemented. The number of pass transistor levels from the

front terminals of the function generator to its output is defined by the number of side terminals r. Shorter transistor chain will thus result in smaller delay.

As it was found that the inclusion of the D flip-flop reduce the overall area [Rose90b], the UPLM is designed to consists of a combinational as well as a sequential part where the sequential part consists of 2 D flip-flops (Figure 3.8). The combinational logic of each UPLM is basically a 4-input function generator which consists of two 3-input function generators in addition to a 2-to-1 multiplexer. The two function generators can be configured independently, giving the user the flexibility of implementing two functions of up to 3-variables each. Alternatively, the two generators are combined through the multiplexer controlled by the 4^{th} input allowing the generation of any function of 4 variables. The outputs of the combinational logic part may either be buffered or used as input to the sequential part of the UPLM. The sequential portion of the UPLM, consisting of two configurable master/slave D-flip-flops is designed to ease the implementation of shift registers and counters. The D flip-flops, in conjunction with the programmable logic interconnect will allow the configuration of other types of flip-flops, e.g., T, JK, or RS flip-flops. The D flip-flops have direct inputs allowing the implementation of these functions.

The clock polarity of the D-FF is controlled by a memory element allowing either rising or falling edge triggering. The clock may be enabled or disabled depending on the content of another memory element. The D flip-flop can be loaded directly from the interconnect lines allowing independent uses of the a combinational and the sequential functions. In such a case the output of the function generator bypasses the sequential portion of the UPLM and the sequential part use the direct input. The UPLM block diagram is given in Figure 3.7 and the function table of the Dff is shown in Figure 3.9

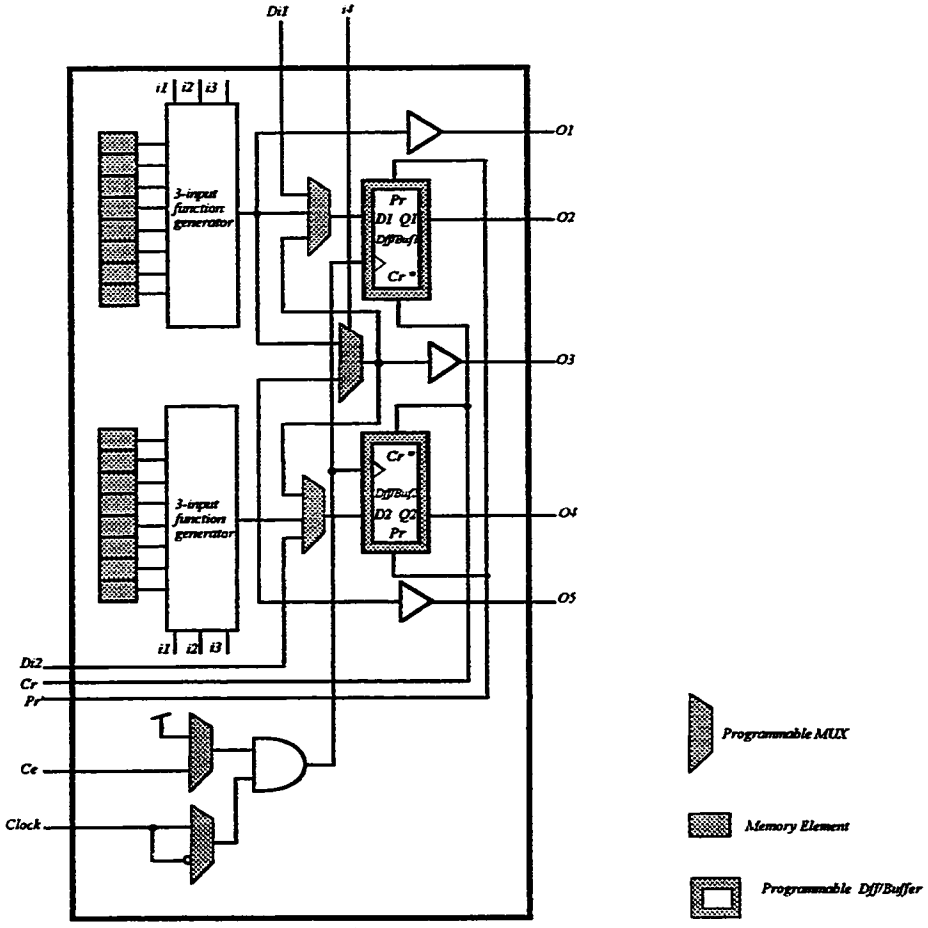


Figure 3.8: The UPLM Logic Block Diagram

Comb/~Seq	Cr	~Pr	O2:O4	Operation
1	0	1	-	Combinational
0	0	1	-	Sequential
0	1	1	0	*
0	0	0	1	*
1	1	x	Not allowed	
1	x	0	Not allowed	

- Output varies with the Intput

* Constant Output

Figure 3.9: The Dff Function table

Figures 3.10 shows the waveform of the output function and an input variable of the function generator implementing the unity function $f(i_1, i_2, i_3, i_4) = 1$. The clock to output waveforms of the generator is given in figure 3.11. A summary of the UPLM performance figures is given in table 3.2.

Table 3.2: The UPLM Performance Figures

Function Generator:

Typical Rise Time: 2.31 ns

Typical Fall Time: 2.31 ns

D Flip-Flop:

Clock to Output Delay: 2.31 ns

Clear to Output Delay : 3.07 ns

Preset to Output Delay: 3.20 ns

Setup Time : 2.80 ns

Hold Time: : 2.31 ns

3.5 Conclusion

The logic module architecture and granularity was chosen for the new FPGA architecture. A table lookup-based logic module was found to be suitable for the static NVRAM programming technology. A universal programmable logic module (UPLM)was designed and verified through SPICE simulation. With a sequential and combinational parts that can function independently, the module also has high functionality. The number of logic inputs of the table lookup was chosen to be four and the number of the UPLM outputs is chosen to be five.

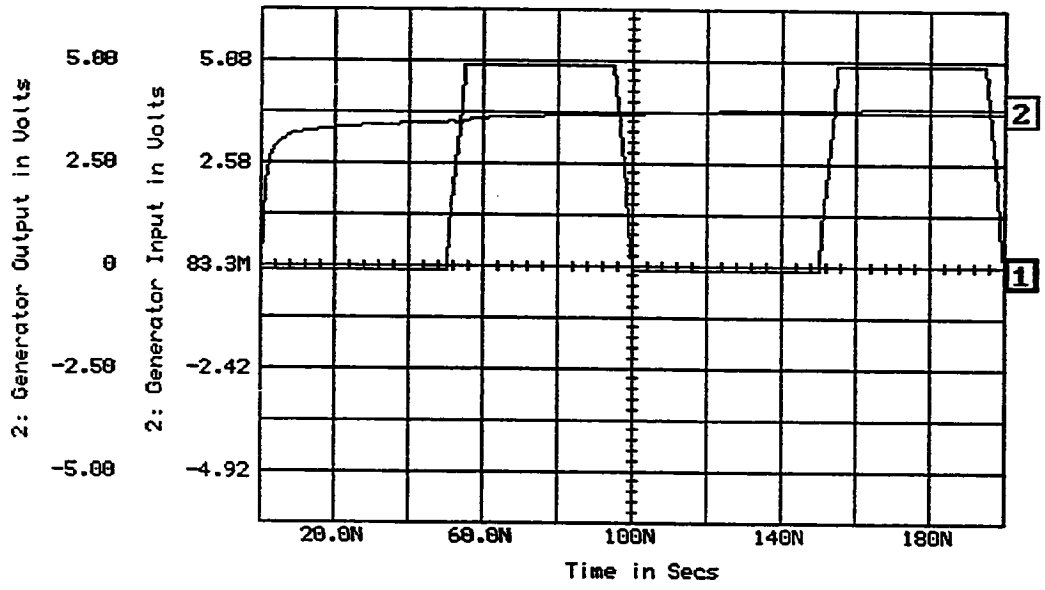


Figure 3.10: The Function Generator Input and Output
($F = o_1/o_3/o_5 = F(i_1 i_2 i_3) = 1$)

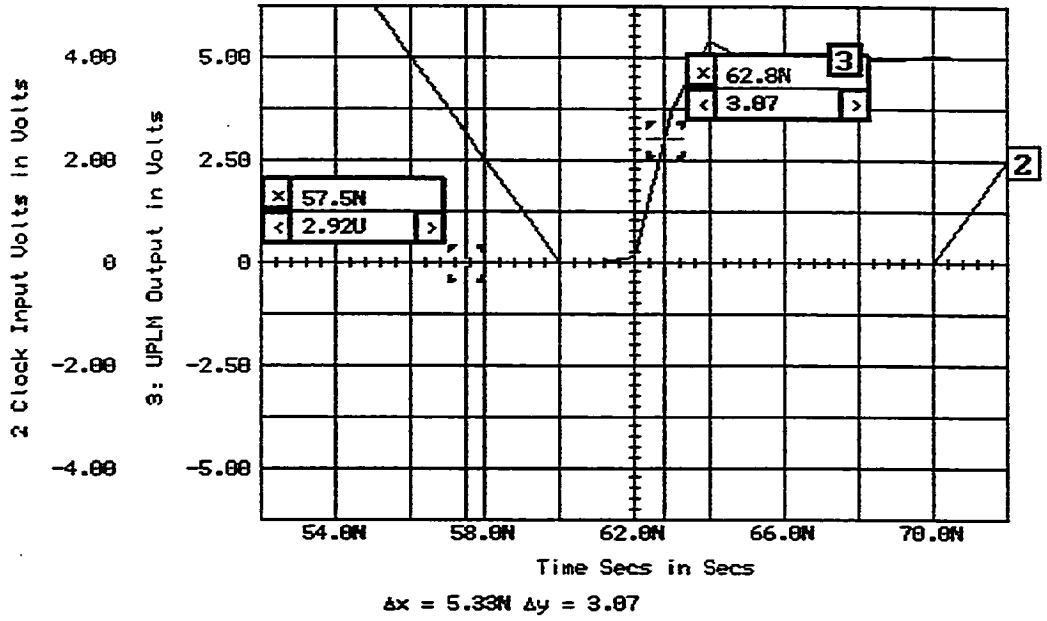


Figure 3.11: Clock to Output Delay of the UPLM

CHAPTER 4

THE INTERCONNECTION STRUCTURE

4.1 Introduction

The general architecture of FPGAs is highly influenced by the array topology and structure. The topology model of a circuit deals exclusively with the circuit connectivity. The structure adds placement and layout of the circuit. The structure of an island style FPGA with vertical and horizontal routing channels is shown in Figure 4.1. Such structure is suitable for the large size non volatile RAM (as compared to antifuse programming element). The UPLMs and input/output blocks in an FPGA are interconnected using a set of switches to form the user logic. Such switches are located at specific locations in the array; namely the switch boxes and connection boxes. The interconnection switches (NMOS pass transistors) which establish the desired paths are controlled by the NVRAM memory cells. In this chapter, the FPGA models used in the definition of the interconnect parameters, e.g. the number of tracks per horizontal and vertical channel, will be presented. The distribution of track segment lengths in each channel will be derived and the flexibility of the switch and connection boxes will be defined. A stochastic topological and structural models have been developed to estimate such FPGA interconnection parameters.

4.2 Interconnect Flexibility

The flexibility of the switch block, Fs, is defined as the total number of possible connections offered to each incoming wire as shown in Figure 4.2. The connection box flexibility, Fc, is defined as the number of channel wires to which each pin of the logic module can be connected (Figure 4.3).

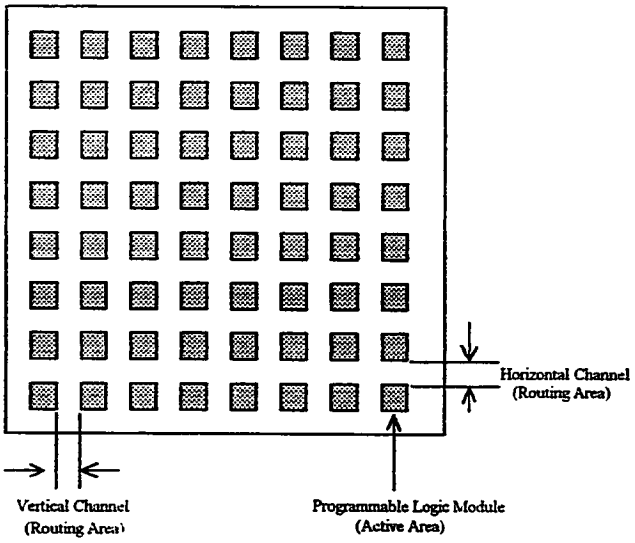


Figure 4.1: The FPGA Array.

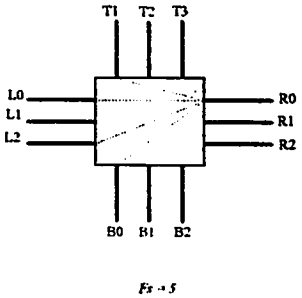


Figure 4.2: The Switch Box Flexibility

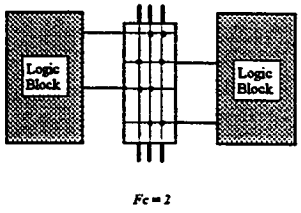


Figure 4.3: The Connection Box Flexibility

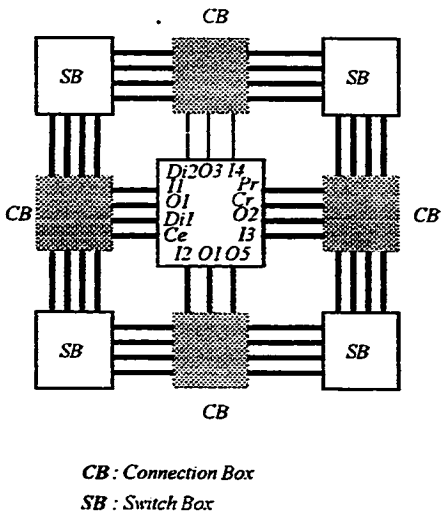


Figure 4.4: Connections of Logic Module i/o's to the Connection Boxes

The connection box, switch box and the logic module relationships are depicted in Figure 4.4. It has been shown [Rose90a] that the minimum number of switches required for full routability occurred with switch box flexibility F_s between 3 and 4. This means that switch boxes need not to be highly flexible. On the other hand, it was found that a high connection box flexibility factor is required for full routability [Rose90a] according the following relation.:

$$0.7 \leq F_c/W \leq 0.9$$

where W is the channel track density. For the routing process of FPGAs to achieve high rate of completion without unnecessarily increasing the area, a tradeoff is required between the flexibility of the switch and connection boxes.

Consider a 2-pin path between two logic modules, which goes through n switch boxes of flexibility F_s (Figure 4.5). Let the flexibility of the connection boxes be F_c. The source logic module pin can connect to F_c different tracks in the source connection box. Each track in the source connection or a switch box can connect to $F_S/3$ since only one pin of the switch box is to be chosen. Then, the total number of different paths between the initial physical pin in the source logic module to a destination block is given by:

$$Paths = F_C (F_S/3)^n$$

In the case where $F_S = 3$, the number of possible paths reduces to

$$\# Paths = F_C$$

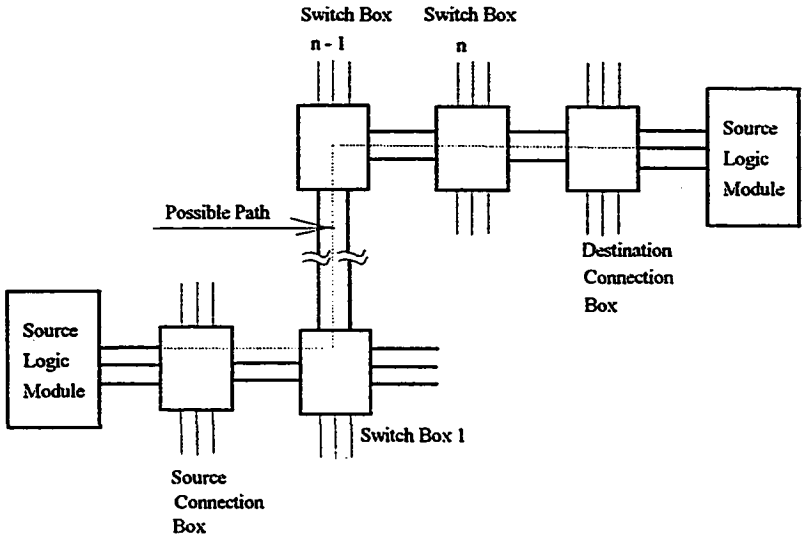


Figure 4.5 : Path Between Two Logic Modules

This value would ease the implementation of the router, since it reduces the choice of possible paths [Bro92].

4.3 The FPGA models

Different logic circuits are implemented in an FPGA by mapping logic to UPLMs. These UPLMs are then interconnected using interconnect segments and programmable switches

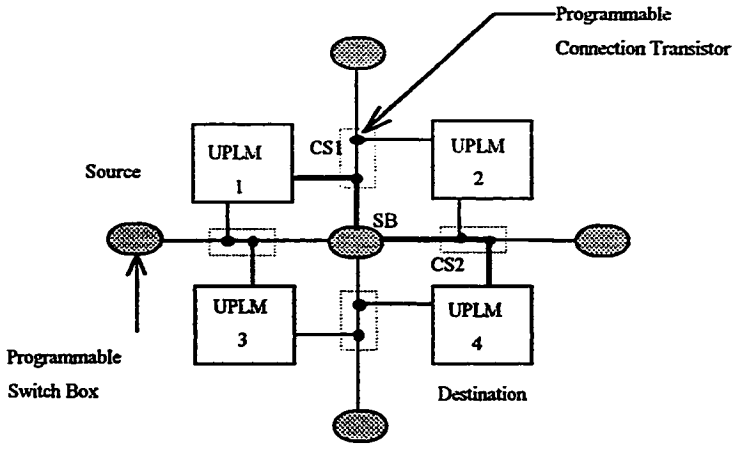


Figure 4.6: Example of WiringNet in an FPGA Array

As an example, Figure 4.6 shows the output of a source UPLM (1) connected to the vertical channel segment by the programmable connection CS1, switch box (SB) changes the direction of the signal to a horizontal channel segment, and a programmable connection switch (CS2) completes the path to the destination UPLM (4).

FPGA models provide mathematical formulations to determine the architectural and the interconnect parameters of the array. These models should abstract the essential features of the array. The topology of a circuit defines how the circuit is connected and can be studied using Rent's rule [Land71]. The structure of a circuit, however, defines the layout of the

FPGA components together with their connectivity. The architecture of a circuit deals with both the topology and the structure. These two factors greatly affect routing flexibility of the interconnection and hence the wiring area, signal propagation delay in the FPGA and the density of the circuit that can be implemented in the FPGA. The topology and structure models of the FPGAs will be defined. These two FPGA models will be used to find the FPGA interconnect parameters.

4.3.1 FPGA Topological Model:

The FPGA topological model consists of a two dimensional grid as shown in Figure 4.7. Each grid point represents a single UPLM. Lines crossing the grid points represent the input/output signals to the modules represented by these points. The vertical and horizontal logic modules center point to center point distances are represented by the grid point vertical and horizontal spacings. The normalizing unit length of a track segment is defined as the distance between two consecutive logic modules and is depicted in Figure 4.8.

4.3.2 FPGA Structural Model

Given the large area of the new NVRAM programming technology, the best structure for the new FPGA is that of an island-like gate array. In this structural model, the FPGA is represented by a two dimensional array of logic blocks separated by horizontal and vertical channels. The channels are composed of track segments of varying lengths. A simplified model for such an architecture is given in Figure 4.9. Track segments are to be programmed for the construction of various nets in the array. Each unit length is a measure of the distance between the centers of two consecutive channels.

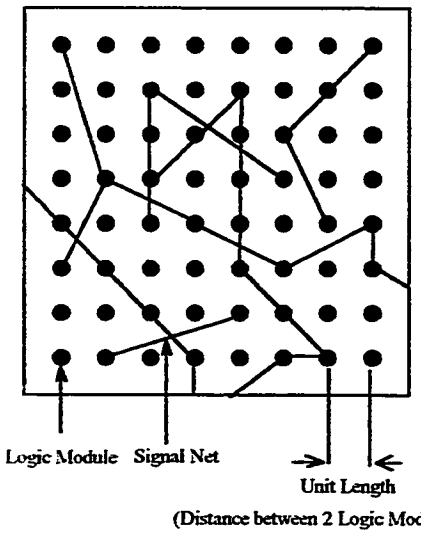


Figure 4.7: The FPGA Topological Model

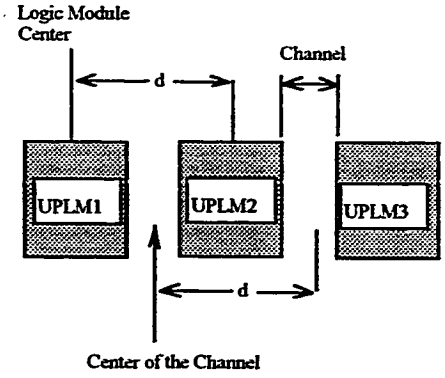


Figure 4.8: Definition of the Normalized Unit Length

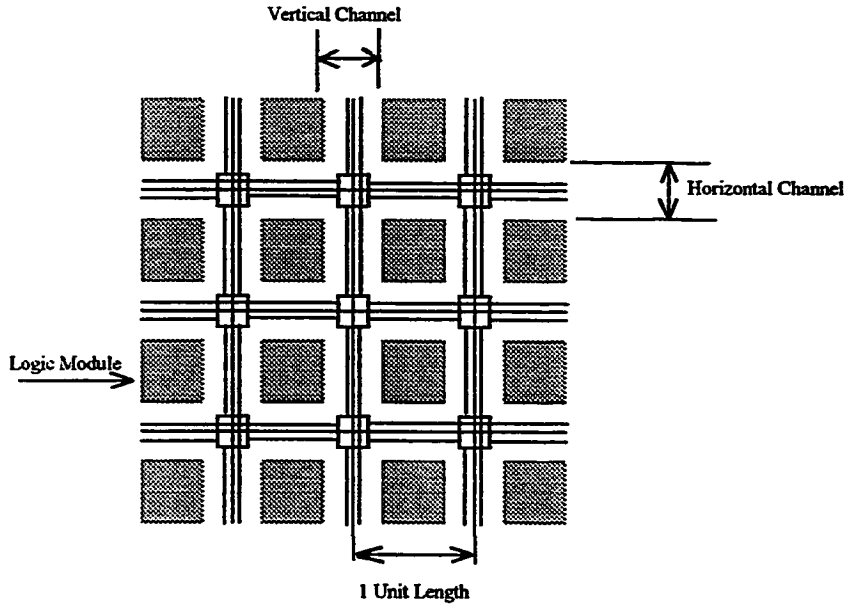


Figure 4.9: FPGA Architecture

The structure of the FPGA represents the floor plan of the array. Before the derivation of the FPGA track density and track length distribution, the interconnect flexibility of the architecture should be defined.

4.4 The FPGA Interconnect Parameters Estimation

In the following, the channel density and track segment length distribution will be investigated. This is done taking into account both the structural and topological models of the FPGA. The topological model allows the use of results from previous work [Feu82]. This model was found for general random logic circuits and is used to find the average wire length in a random logic circuit. It was also used to estimate the wire length distribution within a channel track [Don81]. The structural model will constrain those results to apply to the new architecture taking into account the layout of the components of the FPGA. This model, in conjunction with the topological model, is used to estimate the channel density of the array [Gam81].

4.4.1 Parameters From Topological Model

Various connectivity models to represent logic networks may be adopted. A full connectivity model can be described as a cross bar switch network connecting C blocks, each having A input/output pins. In this case, the required number of switches N_S is given by [Siv88]:

$$Ns = C^2 A^2.$$

The switches need not be clustered, but rather they can be distributed within the array. In this full connectivity model, the number of switches varies quadratically with the number of pins in each logic module as well as with the total number of logic modules. Even though

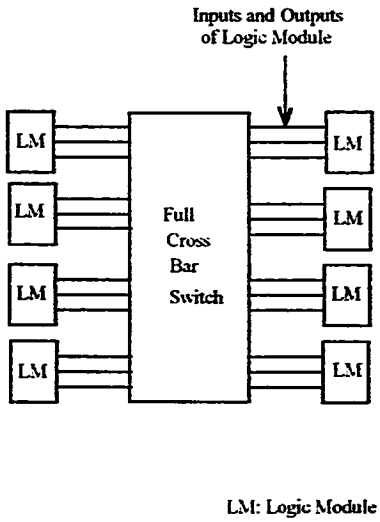


Figure 4.10: Full Cross Bar Switch Structural Model

such a network has high interconnection flexibility, the large number of required connection switches is prohibitively expensive in terms of silicon area.

An alternate, more realistic, model of interconnection is based on Rent's rule [Land71]. Rent's rule is an empirical relationship between the number of wires T which cross an arbitrary boundary containing C logic modules, each having A input/output pins. Rent's rule is given by the following formula:

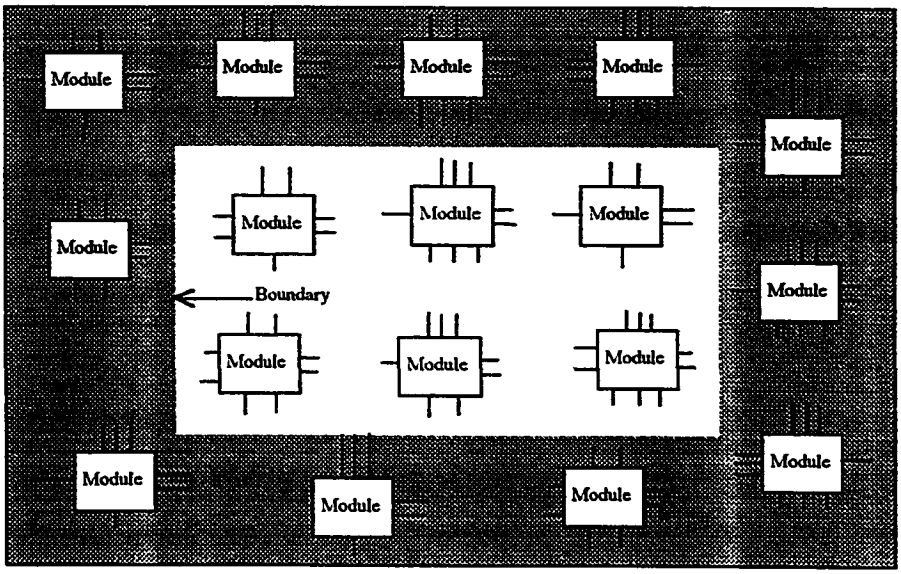
$$T = AC^P \qquad (4.1)$$

The parameter P, known as Rent's constant, is characteristic of the circuit connectivity. Theoretically, $0 \leq P \leq 1$ with higher values of P indicating parallel connection of the logic modules, while lower values indicating serial or cascade connections. Typically, however, the range of P is narrower $0.5 \leq P \leq 0.75$ [Land71]. In this model the circuit is represented as a graph with the vertices, representing the logic modules and arcs representing the

interconnection wires. This relation was also derived theoretically from a stochastic model [Don70,74].

In order to use this model and to help estimate the average track segment lengths, the following assumptions are made:

1. All nets in the array are restricted to two pin nets.
2. The programmable logic modules are modeled as points or zero-area logic modules.
3. The array has infinite dimensions.
4. The distribution of connection wires is a uniform function.
5. The probability of a connection between two nodes depends only on the distance between them.
6. The channel width and the logic module width are equal.



The inside region of the boundary contains C modules (or blocks),
Each block have an average A pins, all the blocks are interconnected (not shown),
the number of wires crossing the boundary is T.

Figure 4.11: Illustration of Rent's Rule

Using Rent's rule under the above assumptions, it has been mathematically shown [Feu82], that the average normalized wire length (in multiples of unit length) is given by:

$$R=2^{1/2}[2.P(3+2P)/(1+2P)/(2+2P)][(C^{P-1/2})/(1+C^{P-1})] \qquad (4.2)$$

where:

P is Rent's constant as defined in equation (4.1),

C is the number of logic modules in the array, and

R is the average normalized wire length

The average wire length values obtained from this formula are reasonably close to those obtained from experiment [Feu82].

Rent's rule was used by [Don81] to find the wire length distribution in random logic circuits. The approximation made in this work is that this distribution is applied to channels of an FPGA rather than to non regular structures. A track segments length distribution function f_k for good two-dimensional placement, which was experimentally verified, is estimated using equation (4.1) and is given by:

$$f_k = g/k^q \qquad (1 \le k \le (C/2))$$
$$\approx 0 \quad (k>(C/2)) \qquad (4.3)$$

where q = 3 - 2P, and f_k is the fraction of wires with length k ($0 \le k \le C/2$, for a CxC array) and g is a normalizing constant satisfying the condition.

$$\sum_k f_k = 1$$

4.4.1 Parameters from Structural Model

This model, in conjunction with the topological model, was used to estimate the channel track densities. The channel width is estimated according to the results in [Gam81] based on the following constraints:

1. The number of wires emanating from every logic module is taken to be randomly distributed according to Poisson distribution with parameter λ.
2. Each wire length is assumed to be independently chosen according to a geometric distribution with mean R. The wire length is assumed to be independent among the different modules.
3. Restricted wiring directions are also .(See [Gam81] for more details).

In this case the expected value of the width of routing channels is:

$$W_{avg} = (R.\lambda/2) \qquad (4.4)$$

where R is the average wire length in the array and λ is the average number of pins emanating from one logic module. Before using these results for the estimation of FPGA normalized interconnect length, we list their effect on the FPGA architecture:

1. The model maps each logic module into a point (zero area), while FPGA logic module has finite non zero area.
2. The nets are assumed to be 2 pin nets, whereas the FPGA may implement circuits with multiple pin nets in general.
3. The array is assumed to be of infinite dimension, whereas real FPGAs have finite array dimensions.
4. The number of wires emanating from each logic block is constant.
5. The programmability of the FPGA interconnect is not taken into account by the model.

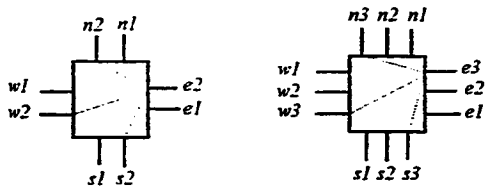
6. The model used to derive equation (4.2) assumes a uniform distribution function for the wire length, whereas the model used derive equation (4.4) assumes a geometric distribution. This does not affect the estimation since none of the two assumed distributions is used in the derivation of the two equations.
7. The model assumes that the width of the channel and that of the logic module are equal, which is generally not true.
8. The models do not take into account the fact that the array is i/o bound, i.e., no restrictions are made on the number of pins of the chip.
9. The models do not take into account the physical properties of FPGA components such as the interconnect capacitance and resistance, and their physical dimensions.

4.4.3 Procedure for the Interconnect Parameters Estimation

The necessary equations for the estimation of FPGA interconnect parametres are given by equations 4.2, 4.3 and 4.4. In the following, the necessary steps to apply those equations to this particular problem are detailed.

The regularity factor is defined as the number of physical transistors in a chip divided by the number of individually designed transistors [Gla85]. Clearly, it is recommended to have a high regularity factor for better design productivity and layout efficiency. High regularity factor can be achieved by having a single design of each FPGA component. The logic modules of the FPGAs are the same all over the array, however, the interconnection structure is irregular within a single channel due to the varying track segment lengths. This will be translated into irregular switch boxes. For a switch box of flexibility 3, 6 pass transistors are needed for each pin coming to the box on one side and leaving in the three other sides. For a switch flexibility of 3, the switch box in Figure 4.12.a the required number of pass transistors is 12, whereas 18 pass transistors are needed for the switch box

in Figure 4.12 b. Obviously an FPGA array having both of these switch boxes will be irregular. The design objective of the interconnect structure is to constrain the theoretical values found using the FPGA models, such that regularity is maintained all over the array. The approach taken to estimate the FPGA interconnect track density and the track length distribution in each channel makes of use equations (4.2), (4.3) and (4.4), according to the following procedure:



a. 2-pins/side Switch Box b. 3-pins/side Switch Box

Figure 4.12: Regularity of the Switch Box

1. Find the Rent constant which maximizes the regularity factor as it will be explained in section 4.4.7.
2. Use equation (4.2) and the connectivity constant found in step 1 to define the average wire length R in the array.
3. Use the result of step 2 and equation (4.4) to find the maximum channel density of the array. The number of pins is the same for each logic block in the array.
4. Use equation (4.3) to estimate the track segment length distribution in each channel.

4.4.4 Track Segments Length Distribution

As an illustration, an FPGA of 8x8 UPLMs will be used, each UPLM is assumed to have 14 pins. In addition to the regularity factor, the choice of track segment lengths in segmented channels is driven by tradeoffs involving physical factors which are not represented in the topological, nor the structural models. These factors include the switch resistance, the capacitances of various wire segments and the connectivity of the circuit to be implemented. These tradeoffs are illustrated in Figures 4.13 and 4.14.

Short segments are useful for connecting nearby logic blocks. A fully segmented channel consists of a set of wire tracks with each track consisting of wire segments of one unit length each. Such segmented channel will penalize long nets by incurring unacceptable delays due to the large accumulated resistance and capacitance of the programmable switches. An alternative approach would be to provide channels with continuous long tracks such that there is enough tracks to accommodate all possible nets. In this case even though the track segment resistance is small , their capacitance is large, and area may be wasted since each net occupies a full track. Finally design of the interconnect structure should be regular enough to ease the automation of the device customization process. For the hypothetical FPGA array of 8X8 logic modules and 14 pins per module, the possible track segments that can be present in a wiring channel are shown in Figure 4.15. Considering the case of an island style FPGA (Figure 4.9) where the array consists of only one type of segments, the regularity of the switch boxes will be studied.

If the array has fully segmented channel with one only unit length segments, the switch boxes (at the intersection of horizontal and vertical channels) will have equal number of lines emanating from them leading to a regular layout. In the case where the array has track segments two units long only, the switch boxes are also regular except for the first and the

last one. In the case of track segments of length 3 units, 6 units or seven units. the switch boxes do not have the same number of pins coming out of them, introducing irregularity in the array. For the case of track segments of length 4 or 5 units the switch boxes are regular except for middle switch box. In conclusion, for this particular case (8x8 array of logic modules), to maximize the switch box regularity throughout the array we should restrict the channel segment lengths to include only 1, 2, 4 , 5 or 8 units (the last value represents long lines which traverse the whole channel without going through any switch box). Further constraints will be imposed to account for the models described earlier.

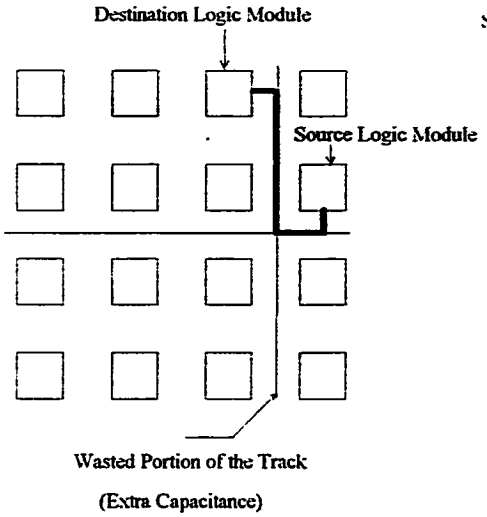


Figure 4.13: A Wiring Net With Long Lines

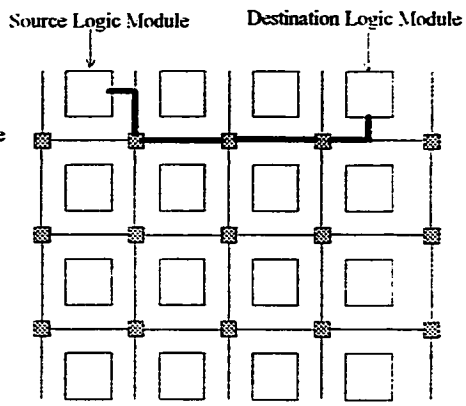


Figure 4.14: A Wiring Net in a Fully Segmented Channel

4.4.5 Connectivity Constant vs Channel Density

Consider the example of 8x8 array and 14 i/o pins per logic module. Varying P from 0.45 to 0.95, it was found that a P increases, the channel density increases as well (Figure 4.16). It can be concluded that for serial circuits (low P), the required channel density is low asexpected. For a highly parallel circuits (High P), the required channel density is high.

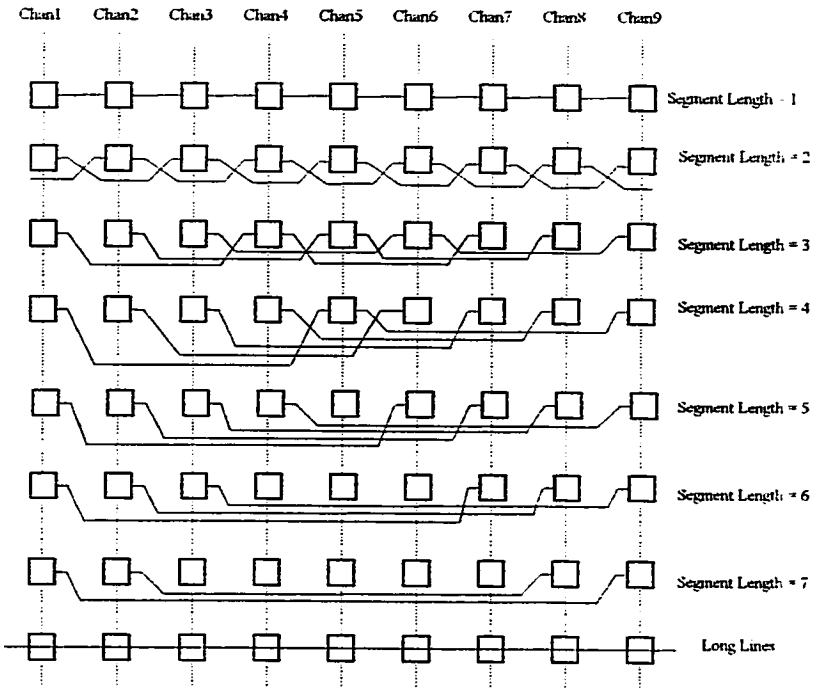


Figure 4.15: Track Segments in a Channel of an 8x8 Array.

4.4.6 Connectivity Constant vs. Track Segments Length Distribution

Table 4.1 gives the channel segments lengths distribution as function of P, which is varied from 0.45 to 0.95 (roughly representing typically reported values). The track segments

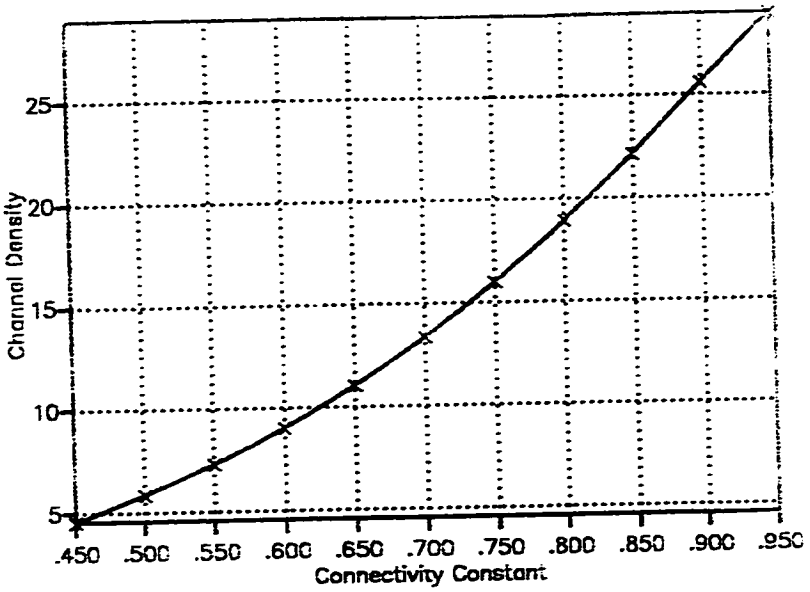


Figure 4.16: Channel Density vs. Connectivity Constant

lengths are varied form 1 to 8 normalized units. The table entries are the percentages of tracks of a given segment length in a channel. The last row of the table represents the percentage of track segments of length greater than 2. The track length distribution is shown in Figure 4.17 for 3 typical values of P (0.45, 0.65 and 0.95).

4.4 7 Estimation of Connectivity Constant

Table 4.1 and Figure 4.17 show that as P is increased, the distribution of channel segments representing the regularity (channel track segments assumed to have one and two unit lengths) is also increased. It is noticed that, the cumulative distribution of the channel segments representing irregularity (channel segments of 3, 6 and 7 unit lengths) increases. For high regularity factor, this type of tracks may be replaced by track types which lead to more regular layout. The fraction of track segments of length greater than 2 is, thus, replaced by track segments which allow regular structure of the switch boxes. An accepted fraction is ≈25% which corresponds to P=0.65 (see table 4.1) leading to a reasonable number of tracks and i/o pins for the FPGA array. This value of P has also been suggested

by Feuer [Feu82]. The fraction of segments leading to irregularity may be replaced either by short segments or by long segments or by a combination of short and long segment tracks. Short segments require additional switches and increase the array area. Long segments are more suitable since they can be used for other purposes such as clock distribution. So the discarded track segments are preferably replaced by long lines.

Table 4.1: Track Segments Distribution vs. Connectivity Constant

Normalized Track Segment Length \ Connectivity Constant →	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
1	67.8	65.47	62.97	60.36	57.63	54.8	51.90	48.93	45.91	42.86	39.82
2	15.8	16.37	16.87	17.33	17.74	18.08	18.35	18.54	18.65	18.66	18.56
3	6.75	7.27	7.81	8.35	8.90	9.45	9.99	10.51	11.01	11.47	11.89
4	3.69	4.09	4.52	4.98	5.46	5.96	6.64	7.03	6.67	8.21	8.67
5	2.31	2.62	2.96	3.33	3.74	4.17	6.49	5.14	5.57	6.21	6.78
6	1.58	1.82	2.09	2.40	2.74	3.12	3.53	3.98	4.47	4.99	5.55
7	1.14	1.34	1.56	1.82	2.11	2.44	2.80	3.21	3.66	4.15	4.68
8	0.86	1.02	1.21	1.43	1.68	1.97	2.29	2.66	3.08	3.36	4.04
>2	16.33	18.16	20.15	22.3	24.63	27.11	29.75	32.53	35.45	38.48	41.6

In conclusion, the interconnection parameters for the FPGA array have been determined. Theoretical and practical assumptions have been made to approximate the Connectivity constant P. Once P is determined, equations 4.2, 4.3 and 4.4 are used to determine channel density, and the track segments length distribution.

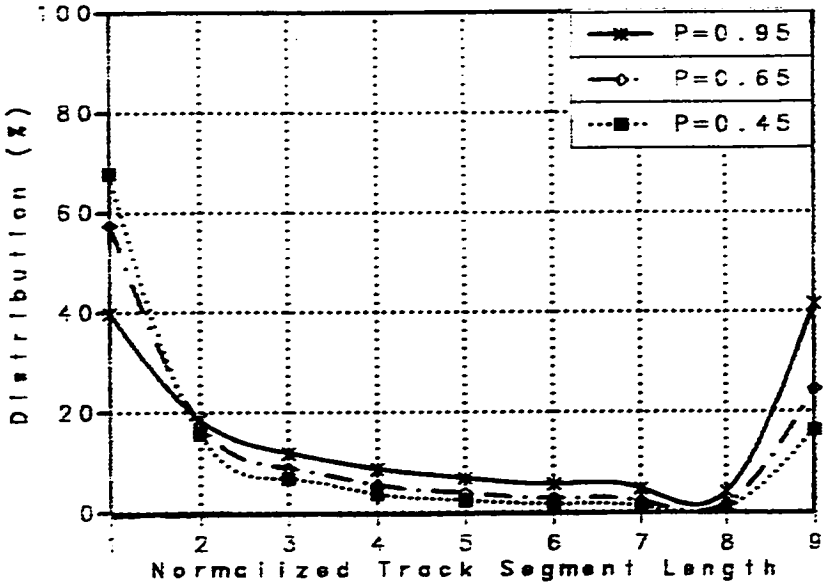


Figure 4.17: Track Segment Length Distribution

4.5 Discussion

In this section, the results from the model used are compared with industrial data [Xil91]. The effect of replacing short segments with long lines are also discussed.

The connectivity constant of an FPGA can be increased to accommodate highly parallel circuits, however, this would increase the irregularity of the interconnect switch boxes, particularly if the number of pins per module is large. Since Rent's rule can be used to estimate the number of i/o pins, the number of chip i/o pins would also increase when P increases. One way to go around this problem is to use a small value of P then scale up the channel density by a certain factor. Equation 4.4 gives the expected value of the width of any channel. The expected channel width will be at least this value and most probably larger [Rose90b]. The modification proposed on to this equation would be:

$$W = [I/2(\lambda R)][\alpha(\lambda, R)]$$

Where α is a non linear function of λ and R. This relation can be seen from the data of actual arrays data [Xil91]. Since R is a function of the number of logic modules in the array C, the factor α depends on C as well.

For those industrial arrays [Xil91], the P constant is found to be small. P decreases as the number of pins of the logic block increases. This is due to the fact that a large P requires a large number of i/o pins. On the other hand, the ratio of the observed (table 4.2) channel density to the calculated channel density increases as the number of pins in the logic module increases. Figure 4.18 and 4.19 show the calculated Rent's constant P versus the number of logic blocks and the number of i/o blocks respectively for the XC3000 arrays. It can be seen that P is almost constant. The maximum relative variation for P is less than 4%. Figures 4.20 and 4.21 show the calculated Ren't constant P versus the number of logic blocks and the number of i/o blocks respectively for the XC4000 arrays. The maximum relative variation for P is less than 8%. Once enough data is available, linear regression could be used to find the relation between the Actual/Estimated ratio and the parameters of the FPGA.

Track segments with length greater than 2 are taken as long lines, this has the effect of increasing the segments capacitance. This increase is given by:

$$\Delta C_i = (8 - i)C_l$$

where i is the track segment length for an 8x8 array (i.e, $3 \leq i \leq 7$) and C_l is the capacitance per unit length of the interconnect. This capacitance increase is less than the capacitance increase when the tracks

are segmented and the switch capacitance is large. Another effect of approximating track segments of length greater than 3 is that these tracks can be used by only one signal path at a time.

The approximation made agrees with the interconnect length distribution for random logic reported by Tewkbury [Tew89] shown in Figure 4.22. Figure 4.23 illustrates the approximation adopted for FPGAs where the medium length segments are ignored and replaced by long lines.

4.6 Conclusion

Mathematical models with the help of some practical assumptions have been used to calculate the FPGA array interconnect parameters, namely the channel densities and the track segments distributions. The idea was illustrated by a hypothetical array of 8x8 logic modules and 14 pins per module. The idea can be extended to arrays of any size. The parameters derived in this chapter, in conjunction with the logic module described in chapter 4, completely defines the architecture designed in this work.

Table 4.2: Xilinx Arrays Parameters

Device	# Logic Modules	# i/o Blocks	# Pins per Logic Module	Estimated P (*)	Estimated Channel Density (Est) (*)	Actual Channel Density (Act)	(Act) /(Est)
XC2064	64	58	6	0.545	3.136	6	1.91
XC2018	100	74	6	0.545	3.387	-(**)	
XC3020	64	64	10	0.446	3.321	7.5	2.21
XC3030	100	80	10	0.451	3.315		
XC3042	144	96	10	0.455	3.299	-	
XC3064	224	120	10	0.459	3.263	-	
XC3090	320	144	10	0.462	3.24	-	
XC4002	64	64	16	0.33	3.602	-	3.33
XC4003	100	80	16	0.325	3.047	12	
XC4004	141	96	16	0.360	2.914	-	
XC4005	196	112	16	0.368	2.802	-	
XC4006	256	128	16	0.375	2.705	-	
XC4008	324	144	16	0.381	2.621	-	
XC4010	400	160	16	0.384	2.547	-	
XC4013	576	192	16	0.391	2.442	-	
XC4016	676	208	16	0.394	2.368	-	
XC4020	900	240	16	0.398	2.74	-	

*: Estimation from the developed interconnect model.

**: No data is available for empty entries.

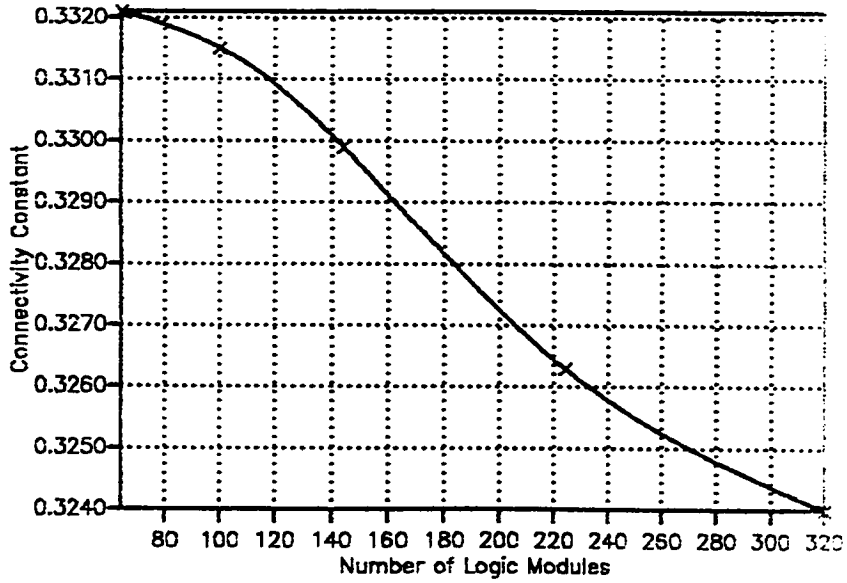


Figure 4.18. Connectivity Constant vs. Number of Logic Modules for XC3000

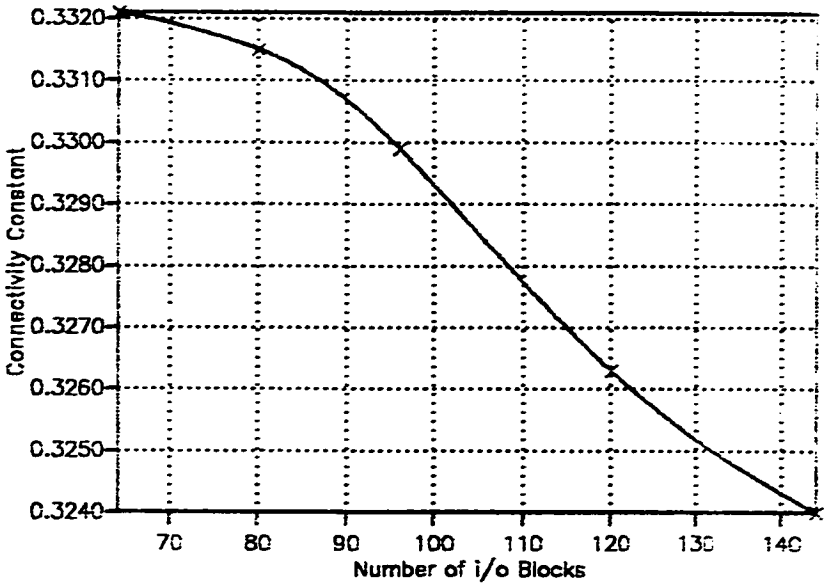


Figure 4.19. Connectivity Constant vs. Number of i/o Blocks for XC3000

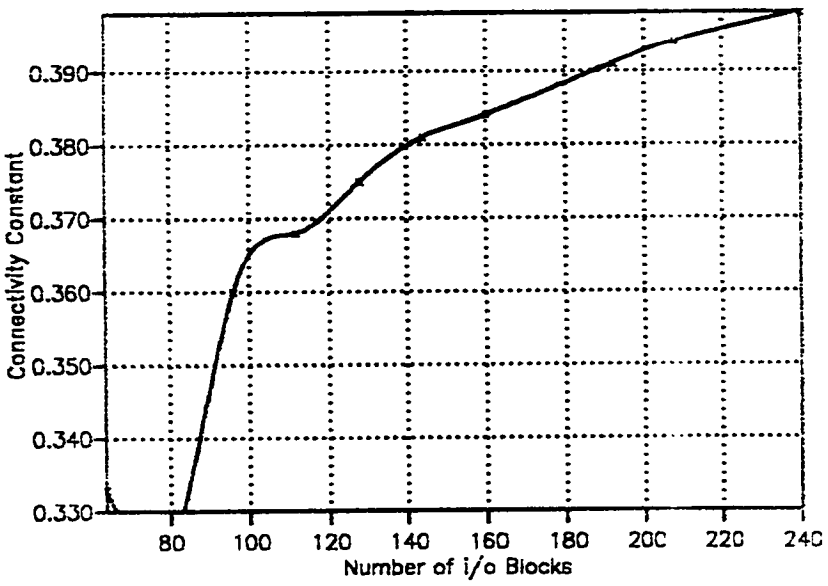


Figure 4.20. Connectivity Constant vs. Number of i/o Blocks for XC4000

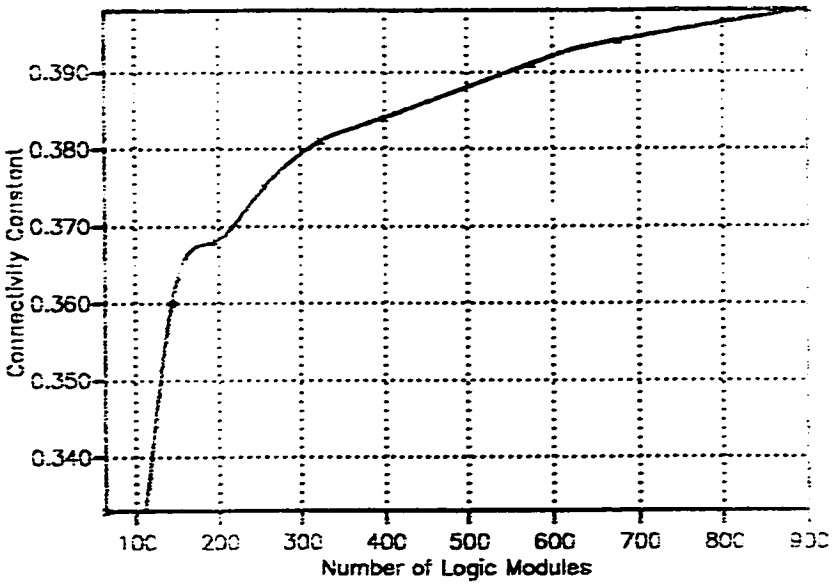


Figure 4.21. Connectivity Constant vs. Number of Logic Modules for XC4000

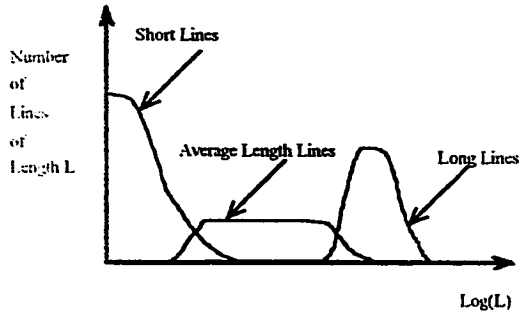


Figure 4.22: Wire Length Distribution in a General Random Logic Circuit

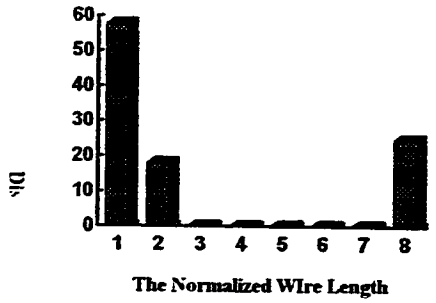


Figure 4.23: Approximation of Wire Distribution in an FPGA

CHAPTER 5

The CLOCK AND POWER DISTRIBUTION SYSTEMS

Among the design issues that have to be addressed in the FPGA design are the clock and power distribution networks. In this chapter the design of the clock and power distribution systems are explored.

5.1. Design Requirements of the Clock Distribution System:

The majority of digital circuits are synchronous. For the purpose of analysis, synchronous designs are modeled as Moore finite state machines. To ensure a deterministic behavior, each closed signal path should contain at least one flip-flop.

The signal flow between various UPLMs in the FPGA array is synchronized by a master clock. The clock is distributed from an external pad to all synchronizing elements (flip-flops) through a distribution network which includes the clock distribution logic and interconnect. The clock serves to unify the temporal design by determining the precise instants of time at which flip-flops change state. This system clock imposes certain

constraints on the timing behavior and performance of the entire FPGA implemented circuits. Hence, it is imperative to properly plan the clock distribution network so as to achieve fast system operation.

A simple FPGA-implemented sequential circuit consists of two registers: a source register and a destination register separated by some combinational logic. A signal path consists of the path through the three elements: the originating register, the combinational logic and the destination register (fig 5.3). The combinational logic performs specific functional manipulation on the signal going from the originating register to the destination register. The clock signal triggers the processed data into the destination register through the combinational network. When designing the combinational network between the two registers care must be exercised to avoid critical path problems, namely the short path delay and the long path delay problems. Paths with short path problems are paths for which the propagation delay is so short that the hold time requirement at the destination flip-flops is violated. Long path problems occur when the propagation delay through the combinational logic is so long that the destination flip-flop setup time requirement is violated. Figures 5.1 and 5.2 illustrate the concept of short path and long path delays.

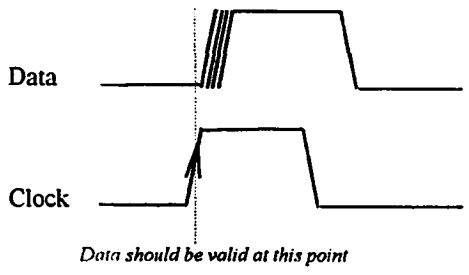


Figure 5.1: Long Path Problems

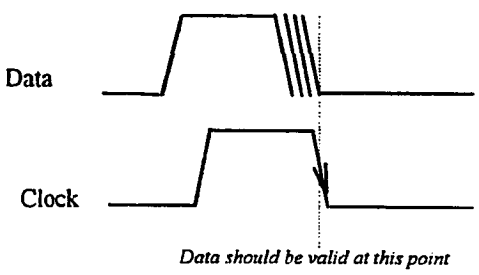


Figure 5.2: Short Path Problems

The delay of a specific sequential signal path consists of six components [Fri86]:

- $c \rightarrow q$: the active clock edge to valid register output Q delay of the originating register in the signal path,
- t_{logic}: signal propagation delay through the combinational logic ,
- t_{inter}: the RC signal delay of the path,
- t_{setup}: the minimum time through the data input of the destination register must be stable prior to the active edge of the of clock ,
- t_{skew} :the time difference between the triggering edges of the processing clock presented to two registers in the signal path. A significant clock skew between the source and destination registers may cause critical race condition,
- $t_{r/f}$: Delay caused by the effect of variation of the input transition time on each transistor within a specific sequential path. For slow rise and fall times, the propagation delay of the device can increase significantly. This delay is described as a time difference since the delay components of a synchronous signal path (the delay components cited previously except the skew and interconnect) change as a function of the input transition time [Fri86].

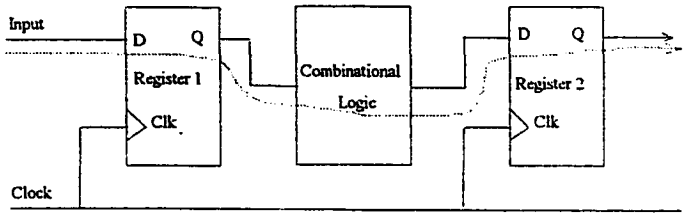


Figure 5.3: An Example of Sequential Path

The total propagation delay from valid register to the valid output of another register in a sequential path (Figure 5.4) is given by:

$$t_{pd} = t_{c\rightarrow q} + t_{logic} + t_{setup} + t_{skew} + \Delta t_{r/f} \qquad (5.1)$$

where t_{skew} includes t_{inter}.

For the clock distribution network to meet the design requirements, the maximum operation frequency f_{max} should not exceed $1/(\max(t_{pd}))$, i.e.,:

$$fmax \leq 1/\max(t_{pd}) \qquad (5.2)$$

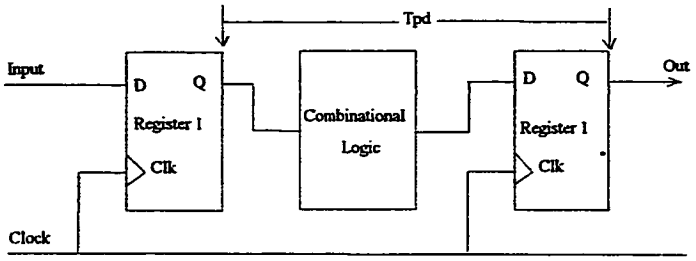


Figure 5.4: Propagation delay in a Sequential Path

In order to increase the performance of the FPGA-implemented circuits, it is necessary to minimize t_{dp}. This can be achieved by decreasing its component delays and in particular nonfunctional parasitic delays. One component delay which is related to the clock distribution network is the clock skew component. Decreasing this parameter will result in better circuit performance.

Consider two UPLMs in an FPGA array as shown in Figure 5.5. The clock skew depends on :

- The RC delay of the clock signal between the source and destination points (or UPLMs).
- The propagation delays through the active elements (buffer or repeaters) if used in the clock paths.

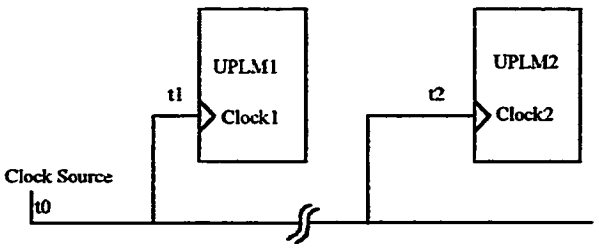


Figure 5.5: Clock Skew Between two UPLMs

The largest component of clock skew is due mainly to the clock line interconnect RC-delay. A good clock distribution network should be used to minimize such skew. One such network uses an H-tree structure as shown in Figure 5.6 [Wan83]. This structure is symmetric (tree branches are equalized). Due to the fact that long lines are not restricted to carry only clock signals, adopting this scheme will reduce the interconnect flexibility of the array.

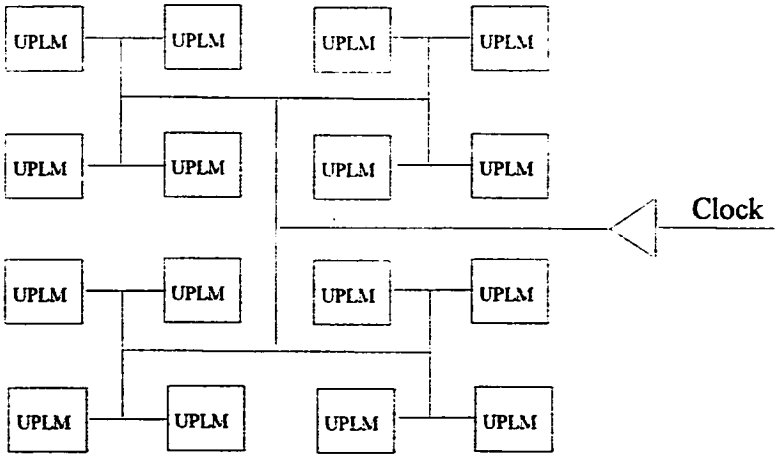


Figure 5.6: H-Tree Clock Distribution Network

This clock distribution scheme, even though suitable for other regular structure, can not be applied to the FPGA arrays due mainly to the programmability characteristics of these arrays. Long lines that run the entire length of the array should be used to distribute clocks so as to minimize the skew. Such lines can carry other global signals depending on the array customization. A clock distribution network suitable for this case is shown in Figure 5.7. If the clock signal enters the array at the middle of the vertical channel as shown in Figure 5.7, the worst case clock skew occurs between the UPLMs located near the clock buffer and the one located farthest from the buffer

.5.2 Estimation of the Clock Interconnect Minimum Width:

The dissipated ac power in the clock interconnect line is given by:

$$P = C.V^{2}.F \qquad (5.3)$$

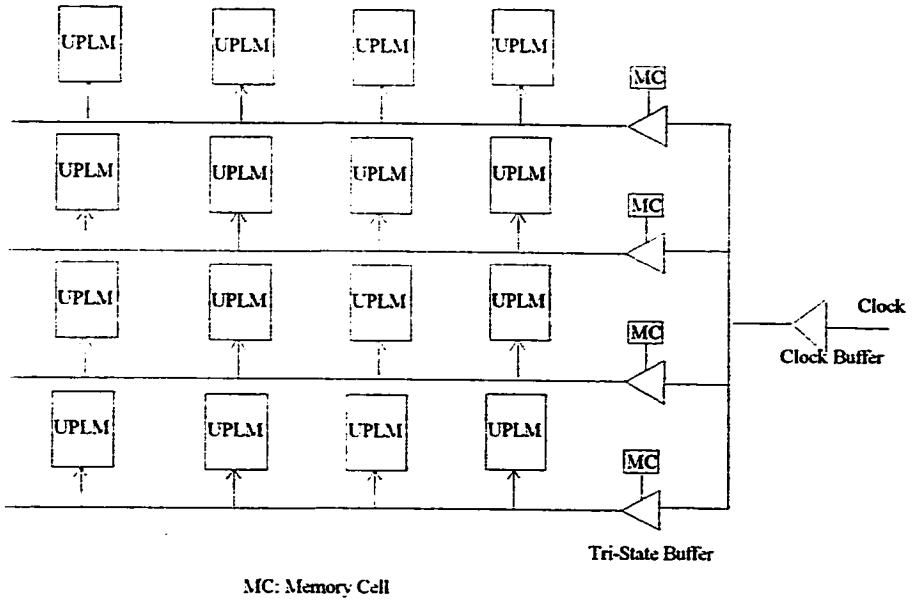


Figure 5.7: Functional Diagram of the Clock Distribution Network

Where C is the total capacitance of the line, V is the supply voltage and F is the clock frequency. The root mean square value of the allowed current is given by:

$$I_{rms} = 2 \cdot C \cdot V \cdot F \qquad (5.4)$$

For signal lines, this is the ac current from the supply during both the charge and discharge of clock cycles [Boon89]. The critical current density J_c through the clock metal line determines the width of the clock line so as to avoid the electromigration effect in the metal conductors. Thus,

$$J_c \geq I/A_c \qquad (5.5)$$

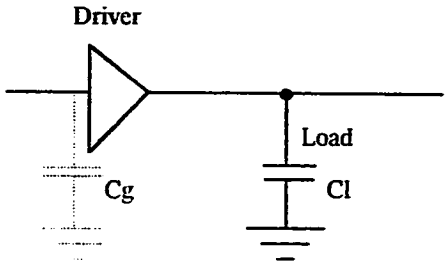
where I is the current flowing in the conductor, and A_c is the critical cross sectional area of the conductor. Thus, from equations 5.4 and 5.5, the minimum metal width should satisfy

$$W \geq 2VCFT_cJ_c \qquad (5.6)$$

where T_c is the critical thickness and W is the metal width required. Only long metal lines will be used for the clock distribution since they incur the smallest RC delay and hence the smallest skew.

5.3 Clock Buffers Design:

The following argument is valid for the design of the clock buffers and for the interconnect long line buffers. The clock interconnect lines represent a significant capacitive load to the clock driver. Buffers are used to drive the capacitive loads. A capacitive load mismatch may occur as shown in Figure 5.8.



Clock Interconnect Driving

Figure 5.8: Clock Interconnect Driving

The driver ac fan out is $Y = C_l/C_g$. If this funout is large the propagation delay may increase to an unacceptable value. If the propagation delay through the driver is t_p, the time taken to charge the capacitive load is Yt_p.The insertion of a buffer of comparable size to the load size will not solve the problem since the driver still have to charge a large load. One approach to solve this problem is to use a cascade of buffers of gradually increasing

sizes [Mead82]. The largest of them will drive the large capacitive load. If the smallest driver have a capacitance of C_g, the next will have a capacitance fC_g, the next will have a capacitance of f^2C_g, the n^{th} will have a capacitance of f^nC_g. f is a factor depending on Y such that:

$$Y = f^n$$

The required number of stages is directly derived from the above formula as :

$$n = ln(Y)/ln(f)$$

The propagation delay in each stage is ft_p. The propagation delay through the n stages is then:

$$t_{pd} = nft_p = ft_p[ln(Y)/ln(f)]$$

The minimum delay through the cascade of drivers is found by taking the derivative of t_{pd} with respect to f. It is assumed that Y is constant. This minimum is found for f = e. The minimum propagation delay through the chain of cascaded drivers is then::

$$t_{pd} = e\, t_p \ln(C_l/C_g)$$

where e is the base of the natural logarithm. Therefore, the required number of stages for minimum delay is:

$$n = ln(C_l/C_g) \qquad (5.7)$$

The steps to design the clock buffers are as follows:

- Design of minimum size driver. The driver sizes are chosen such that the high to low and low to high transition are the same. Also calculate its gate capacitance.
- Find the equivalent capacitance of the network seen by the driver
- Find the number of stages needed to drive the capacitive load using equation 5.7.
- Design each stage.

Long interconnect lines may present a significant resistance to the clock signal. In that case, the repeaters must be used to transform the RC delay in the network to a capacitive load.

5.4 Power Distribution Network

Power distribution is one of the important issues that should be addressed in the chip design. The routing of the power distribution network should be well planed in order to avoid electromigration and heating problems. The conductor size should be properly chosen for that purpose. The size must at least satisfy the limit imposed by the electromigration phenomenon. Factors that influence electromigration rate are: current density, temperature, and crystal structure.

Another factor that has to be taken into consideration in the power distribution conductor size is the voltage drop that may occur if the conductor has a large resistance. This is illustrated in Figure 5.9. The power lines resistance can cause appreciable voltage drops since the current in the power lines is usually larger than that in signal lines. The width of the power distribution network W should be large enough to reduce the power line voltage drop. The effect of the power line drop can be seen from Figure 5.9. The input of the first inverter is 1, the output of the second inverter is supposed to be 1. But since

$$V_{gs2} = V_{ds1} + R_{gnd} I_{gnd} \geq V_{th}$$

a large IR drop may cause an error at the output of the second inverter .

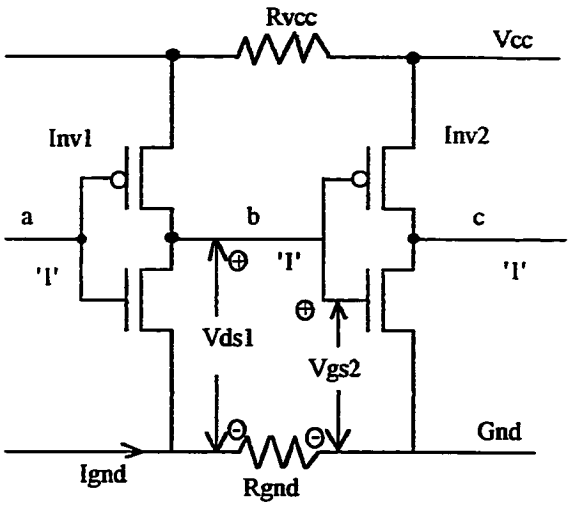


Figure 5.9:The Power Line Drop.

Routing the power network consists of two main tasks: (1) the construction of the interconnect topologies and (2) the determination of the width of various segments in the topologies. The FPGA power distribution network consists of two networks, Vcc network and ground network in addition to Vpp and Vss networks. The following discussion concerns Vcc and ground; the same arguments are valid for other networks. Due to the regular structure of the FPGA the Vcc and Gnd network topologies as illustrated in Figure 5.10. The power network uses metal lines. Polysilicon or diffusion are used only as

"jumper" connection to cross the Vcc lines over the ground lines and can not be used to run power over a long distance because of their large resistance.

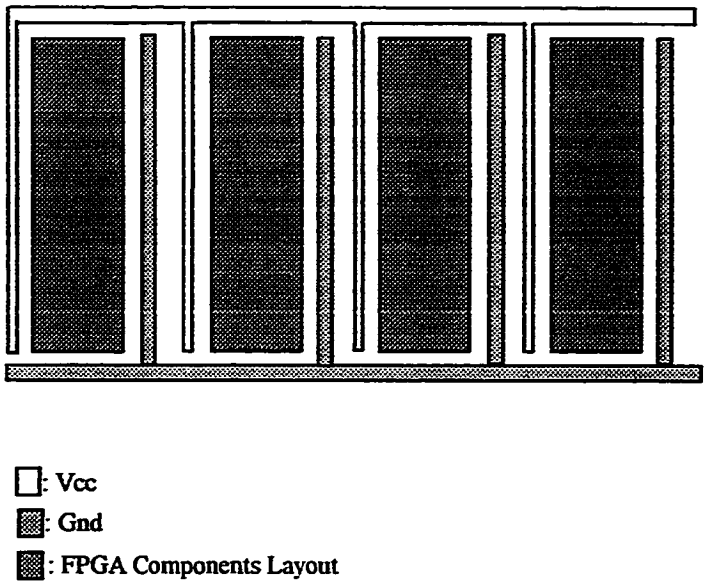


Figure 5.10: Vcc and Ground Networks Topologies

For a normal operation of the FPGA, and since the technology used in the design is CMOS, there is no quiescent power dissipation. There is only switching power dissipation during transitions. For an inverter, when the output switches from low to high, a charge of $Q_h = C_l V_{cc}$ is transferred to the load capacitance. When the output switches from high to low, a charge $Q_l = C_l V_{cc}$ has to be removed from the power supply to ground. If the output switches at a frequency f Hz, the charge transfer to ground is fQ_l per second. The average ac current from Vcc to ground is:

$$Irms = fQl = fC_l Vcc$$

This value should determine the width of the power and ground conductors. The minimum width needed to avoid the electromigration problem is given by:

$$W \geq VCF/T_c J_c$$

From this formula, the conductor size depends on:

- the maximum frequency at which the FPGA flip-flops can be operated,
- the capacitive load, and
- the supply voltage.

A schematic tree of the power network is shown in Figure 5.11. From the formula of the *rms* value of the current, the only factor that depends on the physical dimensions of the interconnect lines is the value of the capacitance load presented to the power signal at the entry of every branch in the tree (C0, C1 and C2 in Figure 5.11). This capacitance has two parts: the interconnect capacitance and the load capacitance in each branch. If the operation frequency is known and the voltage source value is known, the metal width can be calculated depending on the process specification for the electromigration phenomenon avoidance.

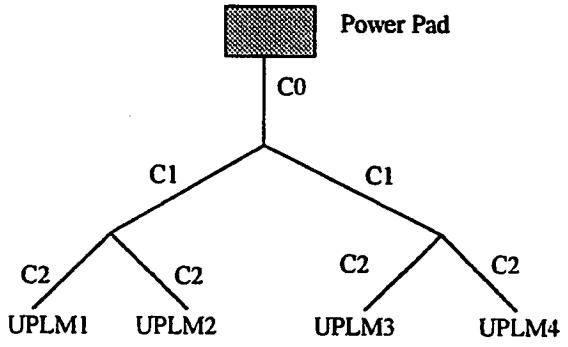


Figure 5.11: A Power Distribution Tree

5.5 Conclusion

The design of the clock and power distribution networks for the FPGA array are crucial for proper operation. Design issues related to these systems have been discussed. One of the critical parameters is this design is the conductor width. The minimum width should satisfy the requirement of the RC delay and skew in the case of clock distribution system and the power current drops in the case of the power distribution system. The minimum width should satisfy the electomigration phenomenon avoidance in both of the networks.

CHAPTER 6

FLOORPLANNING AND CONFIGURATION MEMORY ORGANIZATION

6.1 Introduction:

Floorplanning refers to the task of planning the flow of signals, power bussing and logic placement on the chip. For efficient floorplanning, the dead zones or areas that are usually occupied by the interconnect wires should be minimized. Larger reduction in the total chip area typically results from improved block tiling rather than from individual optimization of individual block areas. The minimization of the interconnect wiring areas can be achieved by:

1. Suppression of pure interconnect zones by cell abutment.
2. Superposition of logic and interconnect, where the logic is implemented beneath the interconnect.
3. Exploitation of the block transparency, where the interconnect is routed through a block rather than around it.
4. Use of tiling.

FPGAs are composed of the following constituent blocks: logic blocks, connection box blocks, switch box blocks, and wiring channels. The general FPGA architecture is depicted in Figure 6.1. Special attention should be paid to the capacitive and resistive wiring parasitics on signal lines. By minimizing wire length between the constituent blocks of the

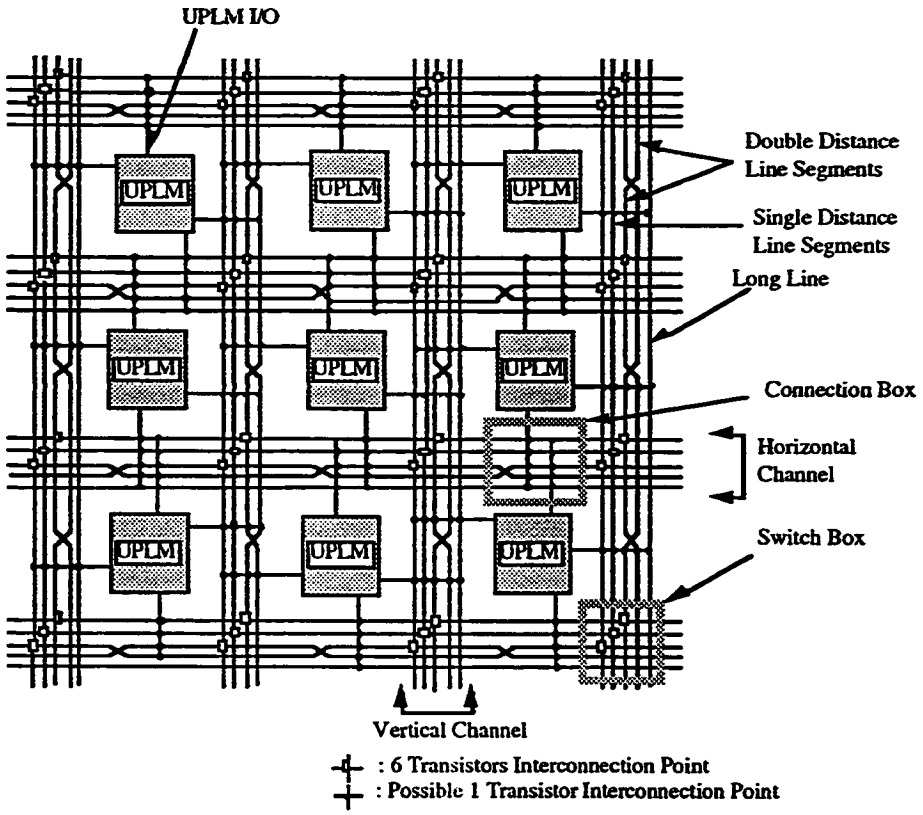


Figure 6.1: The FPGA Architecture

FPGA and within the constituent blocks themselves, both the chip area and the wiring parasitics are reduced, therefore improving performance.

Placement of FPGA modules in the floorplan is straight forward. This is due to the regularity of the array architecture and to its structure.

6.2 Layer Assignment

A CMOS process technology using double metal and double polysilicon layers is assumed. The diffusion and Poly1 layers are used only for the logic and local interconnect. The two metal layers and the Poly2 layer are used for global interconnect. The global interconnect lines carry power, clock, and global signals, e.g., memory cell control signals. The memory cell control signals P and R/W are run horizontally in Metal2, while its control signals S and Data run vertically in Metal 1. The control signal C is run vertically in Poly2, while the clock, Vcc and Gnd lines run vertically in Metal1. The horizontal channel segment lines run in Metal2, while, the vertical channel segments run in Metal1. The principle of block transparency and th superposition of logic and interconnect were used to optimize the layout. The memory cells are distributed all over the chip, and their control signals are driven by control circuitry located at the periphery of the chip.

6.3 The UPLM Floorplan

6.3.1 Memory Cell Layout

Memory cells are distributed all over the core of the FPGA chip. This arrangement allows them to be near the circuitry they control. A conventional memory floorplan would

require unacceptably long connections between the memory cell outputs and the controlled circuitry and the buffering of such signals.

The memory cell control lines cross the entire array either horizontally or vertically. The floorplan of a memory cell is shown in Figure 6.2.

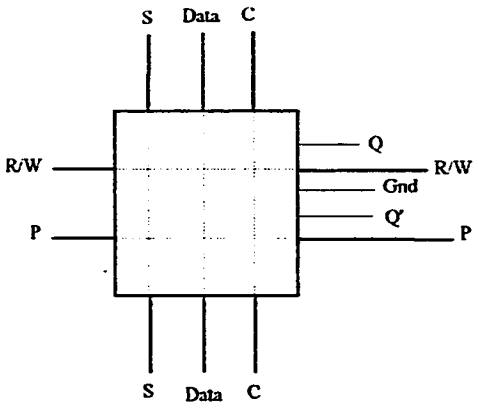


Figure 6.2: A Memory Cell Layout

These lines are shared by all memory cells located on the same row or the same column. The S , data, and C lines are shared by cells in the same column, whereas R/W, and P lines are shared by cells in the same row. The Q an Q' configuration outputs of the memory elements are used by the local UPLM, interconnect switches or the i/o blocks. The Q, Q' and Gnd signals are routed in Poly1 and Metal1 in such a way that they abut within the combinational circuit inputs as well as the sequential circuit inputs.

6.3.2 The UPLM Layout

The functional diagram of the UPLM was given in Figure 3.10.. It consists of 2 parts: a combinational part and a sequential part. The combinational part has two function generator the inputs of which come from table lookup non volatile memory cells. Each function generator uses a table of 8 bit memory cells. The outputs of the memory cell Q and Q' are connected to pseudo inverters which provide the inputs of the function generators. The schematic diagram of the pseudo inverters is shown in Figure 6.4. These devices are used to drive the function generators since the drive capability of the memory cell is very small.

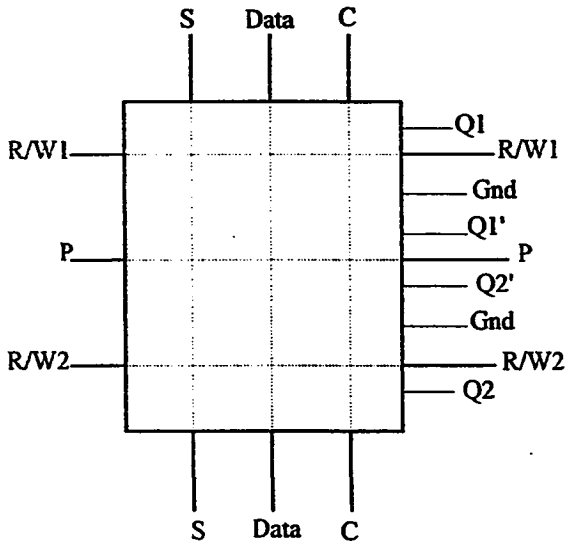


Figure 6.3: Layout for a pair of Memory Cells

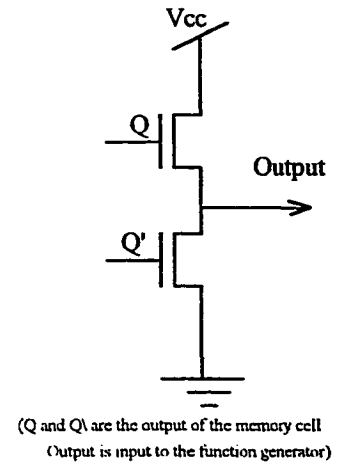


Figure 6.4: Pseudo Inverter

The pseudo inverters share the same Vcc and Gnd lines, which run vertically across the array. The output of the pseudo inverters are abutted to the inputs of the combinational function generator. The general floorplan of the logic module is given in Figure 6.5.

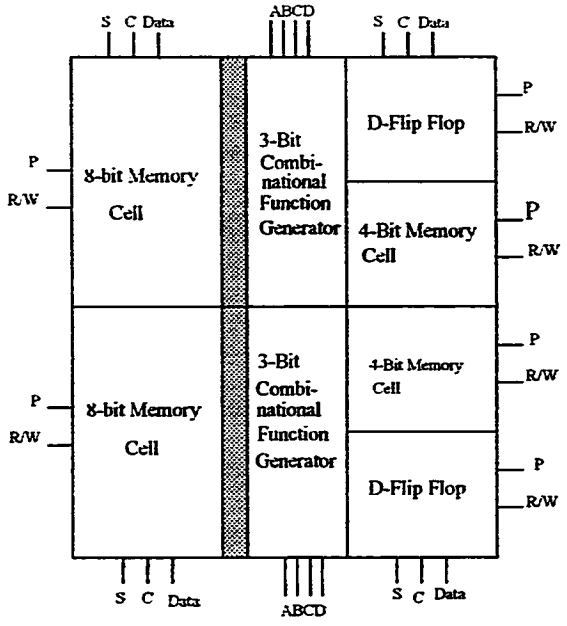


Figure 6.5: The Logic Module Floorplan

The S, C, Data, P and R/W lines are the control lines used in the memory cell. These lines are extended all over the chip. The ABCD signals are the input of the function generators. These lines are extended into the wiring channels to the connection boxes. The outputs of the logic modules (not shown) are also extended to the neighboring wiring channels vertically and horizontally. The clock and power lines (not shown) run vertically. The clear (Cr), preset (~Pr), and clock enable (Ce) signals are global lines coming from the interconnect channels. A set of four memory elements are used to control the clock polarity and activation, the logic module outputs and the D flip-flops outputs.

6.4 The Connection Box Floorplan

The connection boxes connect the outputs and inputs of a logic module to the neighboring channels. This channel can be on one of the four sides: top, bottom, left, and right. The connection box width, including the channel wires, is limited by the UPLM dimensions. The connection box consists of a set of pass transistors controlled by memory elements. The length of the connection box floorplan can be different from that of the UPLM. The memory elements share the control signals with that of the logic modules and switch boxes. The connection box foorplan is given in Figure 6.6.

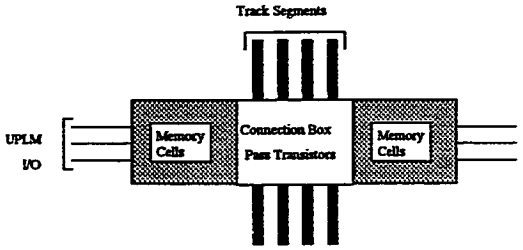


Figure 6.6: Horizontal Connection Box Florplan.

6.5 The Switch Box Floorplan:

The switch boxes change the directions of horizontal signals to the vertical direction and vice versa. A Switch box consists of a set of transistors controlled by memory elements. The memory elements of the switch box share some control signals with the remaining components of the FPGA array (namely the UPLMs and the connection boxes).

The switch box floorplan is given in Figure 6.7. Depending on its location in the array the switch box have different configurations as shown in Figure 6.8. A general floorplan of the FPGA is given in Figure 6.9

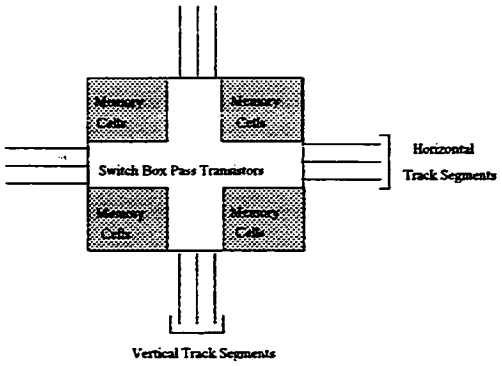


Figure 6.7: The Switch Box Floorplan

6.6 Programming Memory Organization

To minimize wiring area, the configuration memory cells should be laid out close to the various FPGA programmable modules they control (UPLMs, interconnect and i/o clocks). The minimum wiring connections are obtained when the constituent cells and the memory cells are abutted onto the other.

So, unlike the conventional memory organization, the memory cells for the FPGA array should be distributed all over the array. This arrangement is illustrated in Figure 8.10.

The array memory cells are laid out in such a way that they share horizontal and vertical control signals. Such arrangement facilitates the cells programming and erasure. The memory array memory is organized as illustrated in Figure 6.11 where three rows (x, y, and z) and three columns (p, q and r) are shown.

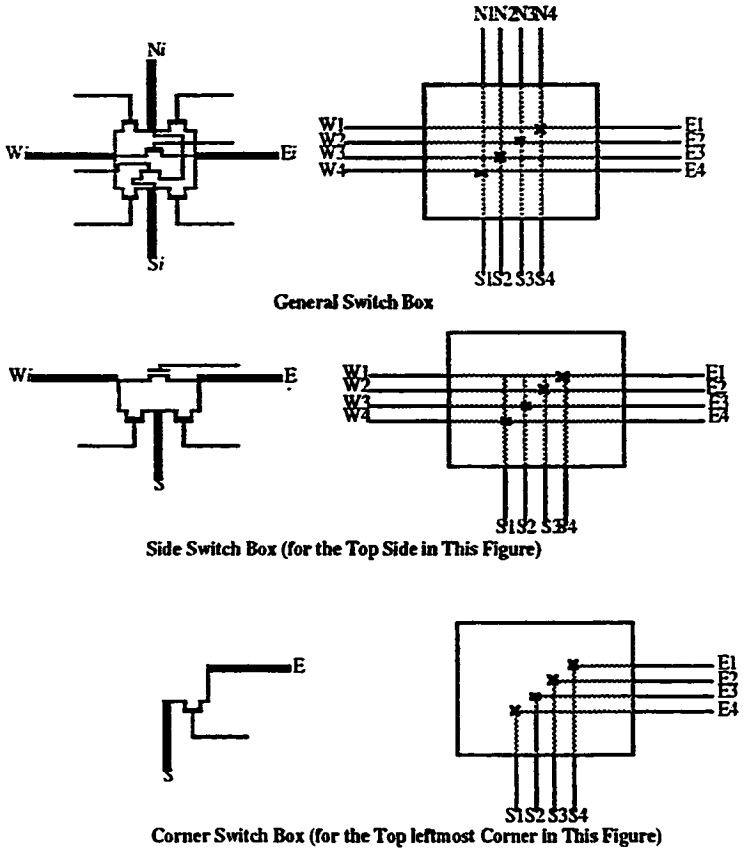


Figure 6.8: Switch Box Configurations

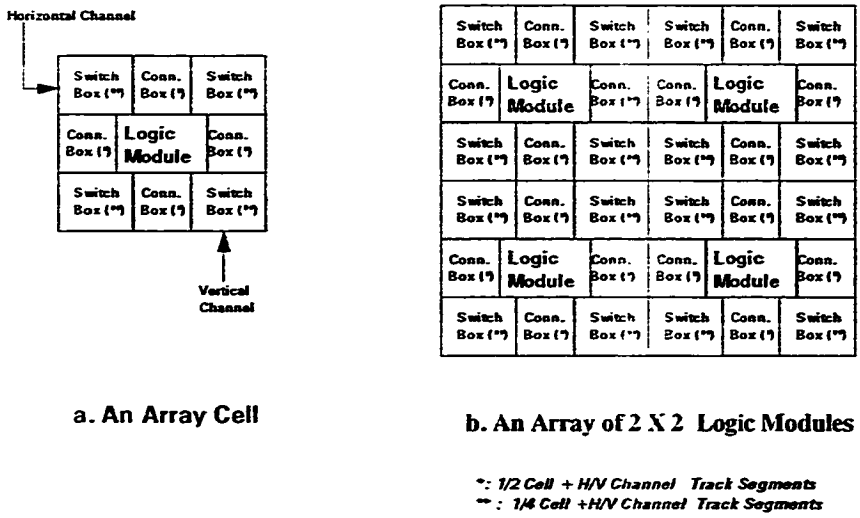
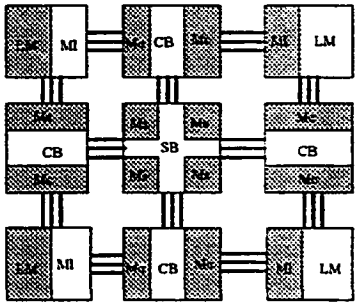


Figure 6.9: The FPGA General Floorplan



LM: Logic Module, CB: Connection Box, SB: Switch Box, MI: Memory cells for Logic Module,

Mc: Memory Cells for Connection Box, Ms: Memory Cells for Switch Box.

Figure 6.10: Functional Memory Arrangement within an FPGA

At the intersection of each of the horizontal and vertical memory cell control lines, there may or may not be a memory cell. The intersection may be empty due to the fact that the number of memory cells in the logic module, in the connection box and in the switch box are not necessarily the same.

The addressing of the FPGA memory cells can be performed as in conventional memory. This requires extra address decoding hardware and external pins. In configuring an FPGA speed of data transfer between the FPGA memory cells is not as critical of a factor as the total chip area and the number of external pins. Only two pins are needed to program the array. Each pin is the input of a shift register. One shift register is used for the cell data, the other is used for address.

The different modes of operation of the memory are selected by the appropriate choice of the address and data words (x, y, z) and (p, q, r) respectively (Figure 6.11). For individual cell operations, the two words must be shifted in each of the data an address shift registers. The data and address words specify which cell will be activated. For sector operation, one row or column is activated. In this case, either the address or data word has to select the row or address respectively. The data or address bits in the shift register will be the same (all 1's). The other shift register content will select the row or column. For flash operation, data and address shift registers will contain the same data bits (all 1's). In this case the operation is done on the entire array. This operation is useful for flash erase of the entire array.

6.7 Conclusion

The general floorplan of the FPGA array has been discussed. The floorplan of the logic module was detailed. The memory organization was given, and the general addressing

scheme was also discussed. Due to its regular structure, the FPGA layout is simplified due

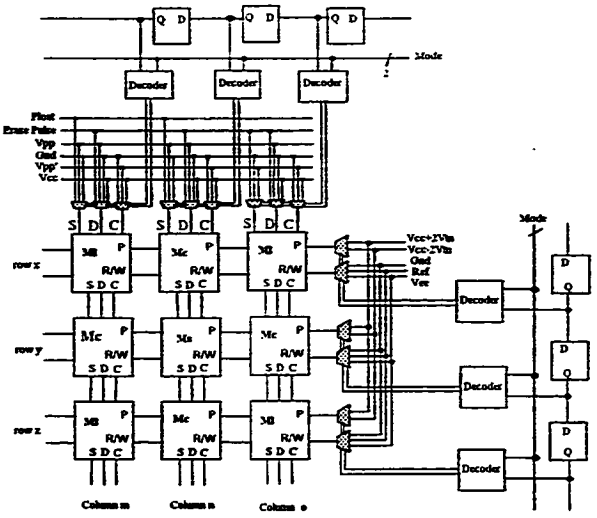


Figure 6.11: Functional Representation of Typical Memory Cells in the FPGA

to the limited number of components that have to be designed. Those components are the connection box, the switch box and the logic module. These components are laid out such that cell abutment is used wherever possible. The FPGA components share control lines running either horizontally or vertically. These guidelines result in reduced silicon area and increased performance.

CHAPTER 7

RESULTS AND SUGGESTIONS FOR FUTURE WORK

7.1. Results

A new FPGA architecture based on a non volatile SRAM technology has been developed. The new architecture enjoys the user programmability and full testability advantages of SRAM-based FPGAs, together with the design security and non-volatility advantages of the antifuse-based FPGAs. It uses a charge-pumped power supply which significantly reduces the ON-resistance of interconnect programming switches. This improves circuit speed and performance as compared to RAM base FPGAs. The new architecture, however, suffers from a larger programming element as compared with the antifuse and the RAM based FPGAs. Furthermore, a more complex process is needed for its implementation as compared to the SRAM based FPGAs. The work reported here represents the initial design phase for this new type of devices. It involved the definition of the overall device architecture including such critical architectural parameters as the logic module granularity and the interconnect structure flexibility parameters as well as wiring channel densities, and wire length distribution. In addition, the layout and simulation of the UPLM has been completed. In summary, the work included:

1. Definition of appropriate architecture for the new static NVRAM programming technology. This has been found to be an island based FPGA architecture.

2. Definition of the basic Universal Programmable Logic Module to be used with this new technology. This included the definition of the logic module granularity, functionality, layout and SPICE simulation.
3. Definition of the required connection and switch box flexibilies.
4. A general procedure was proposed and formulas were developed to determine the wiring channel density and the track segments length distribution within each channel.
5. General guidelines for the design of the clock and power distribution networks were given.

7.2 Suggestion for Future Work

Being a recent technology, several research areas are wide open for future work. This includes:

1. Developing new FPGA architectures both for random logic and for specific applications.
2. Use of FPGAs in the design of new computer architectures such as compute engines.
3. Study of the FPGA testability.
4. Development of various CAD-Tools including design entry, partitioning, logic optimization, technology mapping, and routing.
5. Development of performance driven CAD tools.
6. Investigation of mathematical FPGA architectural models.
7. Investigation of new programming technologies.

Bibliography

References

[Ahr90] Mike Ahrens et al., "An FPGA Family Optimised for High Densities and Reduced routing Delay, "in *Proc. CICC*, May 1990, pp.31.5.1-31.5.4.

[Amin92] Alaaeldin Amin, "A New Non-Volatile Static Memory Cell with Charge-Pumped Power Supply for Use with Application-Specific Reconfigurable Logic, *Pending U.S. Patent.*

[Amin92b] A. Amin, R. Belal and H. Youssef, " A New Nonvolatile FPGA Architecture" in *Proc. ICM'92*, Tunis 1992, pp.4.1.3.1-4.1.3.4.

[Ayat88] Khaled El-Ayat *et al.*,"A CMOS Electrically Configurable Gate Array" in *Proc. IEEE ISSCC*, 1988 pp. 76-77.

[Boon89] Scot Boon *et al.*"High Performance Clock Distribution for CMOS ASICS", in Proc. CICC, 1989, pp. 15.4.1-15.4.5.

[Bro92] Stephen Brown *et al.*, " A detailed Router for Field Programmable Gate Arrays", *IEEE Transaction on Computer-Aided Design*,Vol. 11, No 5, 1992, pp.620,628.

[Car86] William S. Carter et al., "A User Programmable Reconfigurable Logic Array, "in *Proc. CICC*, May 1986, pp. 232-235.

[Don70] W. E. Donath, Stochastic Model of the Computer Logic Design Process", IBM T.J Watson Res. Cent. Yorktown Hights, NY Rep. RC 3136, Nov.5, 1970.

[Don74] ----------, "Equivalence of Memory to Random Logic", *IBM J. Res. Develop.*, vol. 58, No. 5, Sept 1974, pp 401-407.

[Don81] ----------,"Wire Length Distribution for Placement of Computer Logic", *IBM J. Res. Develop.*, Vol 25, No. 3, May 1981, pp. 152-155.

[East91] John East, "Technology Transparent Design", *Computer Design*, December 1991, pp. 25-26.

[EIA87] Electronic Industries Association, "Electronic Design Interchange Format Version 2.00",Washington D.C., May 1987.

[Feu82] Michael Feuer, "Connectivity of Random Logic", *IEEE Transaction on Computers*, Vol. C-31, No. 1, January 1982, pp.29-33.

[Fey87] Curt F. Fey and Demitris E. Paraskevopoulos, " Techno-Economic Assessment of Application Specific Integrated Circuits: Current Status and Future Trends,"in *Proceedings of the IEEE*, June 1987, pp. 829-841.

[Fri86] Eby G. Friedman and Scott Powell, "Design and Analysis of Hierarchical Clock Distribution System for Synchronous Standard Cell/Macrocell VLSI", *IEEE J. Solid State Circuits*, Vol SC-21, No. 2, April 1986, pp. 240-246

[Fur90] Fredrick Furtek et al., " Labyrinth : a Homogeneous Computational Medium," in *Proc. CICC*, May 1990, pp. 31.1.1-31.1.4.

[Gam81] Abbas El Gamal, "Two-Dimensinal Stochastic Model for Interconnections in Master Slice Integrated Circuits", *IEEE Transactions on Circuits and Systems*, Vol. CAS-28, No. 2, February1981, pp.127-138.

[Gam89] Abbas El Gamal et al., "An Architecture for Electrically Configurable Gate Array", *IEEE J. Solid-State Circuits*, vol .24, No.2, April 1989, pp. 394-398.

[Gam91] Abbas El Gamal, Jack Kouloheris,"Tradeoffs in FPGA Architectures, "in *Proceedings of ICM'*91, pp.17-21, December 1991.

[Gla85] Lance A. Glasser and Daniel W. Dobberpuhl, "The Design and Analysis of VLSI circuits", *Addison Wesly Publishing Company*, 1985.

[Gra89] J. P. Gray and T. A. Kean, "Configurable Hardware: A new paradigm for Computation, "in *Proc. Decennial Caltech Conf.* VLSI, C. L. Seitz, Ed., March 1989, pp. 279-295.

[Gup90] Anil Gupta et al., "A User Configurable Gate Array Using CMOS-EPROM Technology, "in *Proc. CICC*, May 1990, pp. 31.7.1-31.7.4.

[Ham88] E. Hamdy et al., "Dielectric Based Anti-Fuse for Logic and Memory ICs,"in *IEDM Tech. Dig.*,pp. 786-789, Dec. 1988.

[Has90] Neil Hastie and Richard Cliff, "The Implementation Of Hardware Subroutines on Field Programmable Gate Arrays, "in *Proc. CICC*, May 1990, pp. 31.4.1.- 31.4..4.

[Hor83] Horowitz, M., "Timing Models for MOS Pass Networks," in *Proc. IEEE Int. Symp. Circuits and Systems*, pp.198-201, 1983.

[Hsi87] , "A second Generation User-Programmable Gate array, "in Proc. CICC, May 1987, pp. 515-521.

[Hsi88] Hung-Cheng Hsieh, "A 9000-Gate User-Programmable Gate Array, "in *Proc. CICC*, May 1988,pp. 15.3.1-15.3.4.

[Hsi90] ----, "Third-Generation Architecture Boosts Speed and Density of Field Programmable Gate Arrays", in *Proc. CICC*, May 1990, pp. 31.2.1-31.2.7.

[Kow90] Keiichi Kowana et al., "An Efficient Logic Block Interconnect Architecture for User-Programmable Gate array, "in *Proc. CICC*, May 1990, pp. 31.3.1-31.3.4.

[Land71] Bernard S. Landman and Roy L. Russo,"On Pin Versus Relationship For Partitions of Logic Graphs", *IEEE Transaction on Computers*, Vol. C-20, No 2, December 1971., pp.1469-1479.

[Rose90a] Jonathan Rose and Stephen Brown, "The Effect of Switch Box Flexibility on Routability of Field Programmable Gate Arrays, "in *Proc. CICC*, pp. 27.5.1-27.5.4, May 1990.

[Rose90b] Jonathan Rose et al., "Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality On Area Efficiency ", *IEEE J. Solid-State Circuits*, vol. 25, No. 5,October 1990,pp.1217-1225.

[Rose91] Jonathan Rose and Stephen Brown, "Flexibility of Interconnection Structure Field-Programmable Gate arrays", *IEEE J. Solid-State Circuits* , vol. 26, No. 3, March 1991, pp. 277-281.

[Sin92] Satwant Singh *et al.*, "The Effect of logic Block architecture on FPGA Performance", *IEEE J. Solid State Circuits*, , vol.27,No. 3, March 1992, pp. 281-287.

[Siv88] Massimo A. Sivilotti, "A Dynamically Configurable Architecture For Prototyping Analog Circuits;"in *Proceedings of he Fifth MIT Conference*, Allen and Leighton, editors, March 1988, pp. 237-258

[Sutan91] S. Sutanthavibul and E. Shragowitz, "Dynamic Prediction of Critical Paths and Nets for Constructive Timing-Driven Placement," *Proc. 28th. Design Automation Conf.*, pp. 632-635, 1991

[Tew89] Stuart K. Tewksbury, "Wafer-Level Integrated Systems: Implementation Issues", *Klewer Academic Publishers*, pp.313-345, 1989, pp.23-73.

[Wan83] Donald F. Wann and Mark A. Franklin, " Asynchronous and Clocked Control Structures for VLSI Based Interconnection Networks", *IEEE Transactions on Computers*, Vol. c-32, No. 3, March 1983, pp. 284-293.

[Whit82] Sterling Whitaker, "Pass Transistor Network Optimize n-MOS Logic", *Electronics*, September 22,1982, pp. 144-148.

[Won89] Sau C. Wong et al., "a 5000-Gate EPLD with Multiple Logic and Independent Interconnect Arrays, "in *Proc. CICC*, May 1989, pp. 5.8.1-5.8.4.

[Xil91a] XILINX, "The programmable Gate Array Data Book", Logic- Drive San Jose- California 95124,1991.

[Xil91b] XILINX, "The XC4000 Data Book", Logic Drive - San Jose- California 95124, 1991.

[Yau70] Stephen S. Yau, and Cavin K. Tang, " Universal Logic Modules and Their Applications", *IEEE Transaction on Computers*, Vol. C-19, No. 2 , pp. 141-149, February 1970.

[You90a] Habib Youssef, "Timing Issues in Cell-Based VLSI Designs", *PhD Thesis*, Computer Science Department, University of Minnesota, Jan. 1990.

[You90 b] Habib Youssef and E. Shragowitz, "Timing Constraints for Correct Performance," *Proc. ICCAD'90*, pp. 24-27, 1990.

VITA

After graduation from high school, RAFFED Belal attended the Faculty of Exact Sciences at University of Sidi Bel Abbes, ALGERIA, then he moved to the "Institut National d'Electricite et d'Electronique", INELEC, Boumerdes, Algiers where he obtained the "Diplome d'Ingenieur d'Etat" in Electrical Engineering, Computers option in April, 1984. In December 1989, he joined the Department of Computer Engineering at King Fahad University of Petroleum and Minerals, Dhahran, Saudi Arabia as a Research Assistant.